GESIS

ZUMA

NACHRICHTEN

Spezial Band 5

A review of software for text analysis

Melina Alexa & Cornelia Züll

Zentrum für Umfragen, Methoden und Analysen (ZUMA)
ZUMA ist Mitglied der Gesellschaft Sozialwissenschaftlicher Infrastruktureinrichtungen e.V. (GESIS). Die GESIS ist eine Einrichtung der "Wissenschaftsgemeinschaft Gottfried Wilhelm Leibniz" (WGL).

Vorsitzender des Trägervereins ZUMA e.V.: Prof. Dr. Max Kaase
Direktor: Prof. Dr. Peter Ph. Mohler

Hausanschrift	**Postanschrift**
B 2, 1	Postfach 12 21 55
68 159 Mannheim	68 072 Mannheim

Telefon	0621/1246 - 268	
Fax	0621/1246 - 100	
E-Mail	zuell@zuma-mannheim.de	
Internet	http://www.social-science-gesis.de/	*GESIS*
	http://www.zuma-mannheim.de/	*ZUMA*

ISBN 3-924220-16-6

Druck: Verlag Pfälzische Post GmbH, Neustadt/Weinstraße.
Gedruckt auf chlorfrei gebleichtem Papier.

CONTENTS

Acknowledgements 5

Abbreviations 6

Part 1: Software for text analysis 7

1. What are the aims of the review 7
2. Who could benefit from this review 7
3. Categorisation of text analysis software 8
4. Concrete selection process 11
5. How the review is organised 12

Part 2: Reviewing the territory 14

1. AQUAD 14
2. ATLAS.ti – The Knowledge Workbench 21
3. CoAn 35
4. Code-A-Text MultiMedia system 39
5. DICTION 4.0 48
6. DIMAP, MCCA and MCCALite 53
7. HyperRESEARCH 61
8. KEDS 69
9. NUD*IST 4 76

10. QED ... 88
11. TATOE ... 92
12. TEXTPACK ... 103
13. TextSmart ... 108
14. WinMAX97Pro ... 116
15. WordStat ... 125

Part 3: Reviewing the reviews ... 133

1. Importing text data ... 135
2. Creating categories and coding text ... 139
3. Exporting data ... 143
4. Linguistic processing ... 147
5. Exploring text data and coding ... 150
6. User interface ... 152
7. Integrating modes of and methods for analysis ... 153
8. Generality of use ... 155

Concluding remarks ... 156

Appendix ... 158

Glossary ... 164

References ... 171

Acknowledgements

We would like to thank the following:

Alan Cartwright, Udo Kuckartz, and Thomas Muhr for kindly supplying us with CD-ROMs of and documentation to their software.

SPSS for kindly supplying us with a free version of TextSmart.

Alfons Geis and Lothar Rostek for reading and their invaluable comments on earlier and later versions of this review.

Hilde Bruins, Alan Cartwright, Ann Dupuis, Udo Kuckartz, Ken Litkowski, Don MacTavish, Thomas Muhr, Normand Peladeau, Lyn Richards, Philip Schrodt, for reading, commenting, and correcting earlier versions of the reviews of their software. The communication with you has been exciting and stimulating, made us question some of the things we took for granted, showed us aspects we had not seen till then, and has been encouraging for our work.

Nanny Wermuth, John Bradley, and Peter Ph. Mohler for reading parts or the whole of the review and for their helpful hints and comments.

Terri J. Hennings for reading and correcting the text. Patricia Lueder for her sharp eye for detail and mistakes, her inexhaustible energy in formatting and the countless coffees and jokes to keep us going.

Last but not least, ZUMA for encouraging and supporting this project.

Abbreviations

AQUAD – Analysis of QUAlitative Data

BCOW coding scheme – Behavioral Correlates of War

C-score – Context score

CoAn – Content Analysis

DIMAP – DIctionary MAintenance Programs

E-score – Emphasis score

GUI – Graphical User Interface

HTML – HyperText Markup Language

ID – text unit identifier

KEDS – Kansas Event Data System

KWIC – Key Word In Context

MCCA – Minnesota Context Content Analysis

MECA – Map Extraction, Comparison and Analysis

*NUD*IST* – Non-numerical Unstructured Data Indexing Searching and Theorizing

PANDA project - Protocol for the Analysis of Nonviolent Direct Action

PC-ACE – Program for Computer Assisted Coding of Events

PLCA – Program for Linguistic Content Analysis

RPN – Reversed Polish Notation

SAS – Statistical Analysis System

SGML – Standard Generalized Markup Language

SPSS – Statistical Package for the Social Sciences

TACT – Text Analysis Computing Tools

TATOE – Text Analysis Tool with Object Encoding

XML – eXtensible Markup Language

WEIS event coding scheme – World Events Interaction Survey

PART 1: SOFTWARE FOR TEXT ANALYSIS

1. What are the aims of the review

This book provides a review of a selection of software for text analysis. Its aim is to provide a detailed and, as far as possible, up-to-date account of the variety of available software as well as catalogue the kinds of support the reviewed software offer to the user. An additional goal is to record the tendencies both in functionality and technology and identify the areas where more development is needed. For this reason, the selection of software reviewed here comprises not only fully developed commercial and research programs, but also prototypes and beta versions. An additional aspect with regards to the kinds of software reviewed is that both qualitative and quantitative-oriented types of research are included. Depending on research purposes and project design the text analyst can profit from available tools independently of their orientation. Today it is often the case that in computational support, the borderline between quantitative and qualitative methodologies can become 'obscure'; instead, one can detect a number of commonalities which can be placed within a broader text analysis context.

2. Who could benefit from this review

We assume that the readers of this book are already familiar with computer-assisted content and text analysis and possess experience in using text analysis software. We believe that both users and developers of text analysis software can benefit from this review: the former being text analysts in both academia and industry. This book may assist users of text analysis software not only in choosing among available programs, but also in testing, enriching and thinking about their analysis methodology. It is not, however, the purpose of this book to provide an introduction or an in-depth discussion of text analysis methods.

Developers of text analysis software may also benefit from the present review, since it documents the variety of how text analysis methods are applied, discusses the differences and similarities between software, and points to some areas for future development.

3. Categorisation of text analysis software

At present, there is a fairly rich variety of software supporting text analysis tasks within different disciplinary contexts in considerably different ways. There are qualitative-oriented as well as quantitative-oriented software. There is also a variety of stylistic, literary and more general text analysis software currently available. These all aid the analysis of text data to a greater or lesser extent by providing means for

- managing texts and their coding
- examining how frequently and how words are used in context as well as exploring the coding, e.g. how often particular categories have been assigned to a word or text segment, which categories and how often occur together, what links or relations exist between categories or coded text segments, etc.
- creating and maintaining categories and categorisation schemes
- assigning one or more categories/codes to word strings, words, phrases, sentences, lines, paragraphs or whole texts
- keeping notes (memos) on text, categories, coded text segments
- obtaining different 'views' of the text data as well as the coded parts of a text or a group of texts
- exporting the coding for further processing with other software as well as generating reports on the performed analysis and
- supporting team or co-operative work for a text analysis project and merging coded texts.

Software for text analysis may be categorised according to numerous criteria: for instance, according to the operating system programs run on or the programming language used for their implementation, according to whether they are commercial products for general use or have been developed within academia for specific text analysis research purposes, or according to their suitability for a particular disciplinary field. Judging from the available literature on the subject (see Fielding/Lee 1991, Tesch 1990, Tesch 1991, Weitzman/Miles 1995, Evans 1996, Klein 1997a, Roberts 1997a among many) two broad categorisations can be detected: programs are either grouped together according to their functionality focus, e.g. database managers, archiving programs, text searching programs, text retrievers, taggers, code-and-retrieve programs, etc., or according to the type of research orientation they support, e.g. quantitative, qualitative, literary analysis.

The categorisation we use for the review draws a line across the above two, covering software packages which support different types of research and which are also characterised by *a combination of functionality.* The reviewed software are not solely text retrievers, or text exploratory tools, but rather they support text analysis by a *combination*

of operations for coding as well as searching and retrieving text and maintaining schemes and dictionaries for coding.

In particular, operations which are of 'primary' interest during the process of analysing text may be organised into four quick-and-dirty groups:

- *Text import and management*: Some of the operations concerned are importing, opening and saving a text or a number of texts, grouping texts together and managing different text groups, retrieving single texts or text groups, merging coded texts and projects and types of text structure recognised or required by the programs.
- *Exploration*: Operations here cover not only text exploration, i.e. information about words or strings of words of a text, but also exploration of the existing coding, i.e. querying and retrieving information about the categories used or the coded text segments. For text exploration purposes some of the operations involved are generating and/or maintaining a list of all words of a text corpus and counting their frequency of occurrence, creating a word index (text source included), general text counts about word types, tokens, sentence length, etc., searching and retrieving occurrences of particular stems, words and strings of words, proximity searches, Boolean operator-based searches, providing Key-Word-In-Context (KWIC) concordances. Some of the operations for exploring the existing coding concern generating and maintaining a list of all coded segments, Boolean and semantic operator-based searches, KWIC displays for one or more coded segments, perhaps grouped or filtered according to a particular scheme category, and linking and hyper-linking coded segments perhaps with typed relations.
- *Dictionaries, categorisation schemes, and coding*: This group comprises such operations as creating a categorisation scheme or a dictionary to be used for coding, and/or importing a scheme or a dictionary, maintaining it, defining relations holding between the categories of a scheme, coding automatically, interactively or manually whole texts and/or specific text segments, assigning notes to texts as well as codes or coded text segments, and merging coding.
- Finally, the group for *export* operations: For example, saving the text only or the coding only or both as well as the categorisation scheme(s) or the dictionary used in different ways, e.g. raw ASCII/ANSI- or HTML- or Standard Generalized Markup Language (SGML)/eXtensible Markup Language (XML)-encoded files, exporting coding to other software packages, e.g. for statistical analysis, and exporting text to database software.

Naturally, text analysis programs differ according to whether or not they support all the above-listed groups of operations. For example, one can distinguish between those text analysis programs which support text exploration from those which do not. More importantly, however, programs differ according to *how* they support a certain operation - all

the programs reviewed here support coding, but the way this is realised or the definition of the coding unit are not the same for each program.

The list of operations is not exhaustive. There is a large number of operations not catalogued above which, depending on each individual text analysis project, can also be considered primary. This would include such operations as managing and coding multimedia material or support for the transcription process or audio material. Clearly, not every single operation of each group and every single group may be required for all projects for each analysis phase, nor has every operation the same importance for all projects.

Still, during the entire length of a text analysis project a combination of operations which belong to the coding group with the operations of at least two of the remaining three groups are required. This claim is based upon four different factors. First, on the gathered user needs as resulted from a survey (Zuell/Alexa 1998) carried out among the users of the computer-assisted content analysis program TEXTPACK (Mohler/Zuell 1998); second, on our daily communication with and consulting of researchers designing and carrying out text analysis studies at ZUMA; third, on the main tendencies and practices in computer-assisted text analysis methodology till today as reported in (Alexa 1997); and last but not least, on the state of the art in text analysis methodology at the moment.

Therefore, we reviewed only those software packages which provide support for a combination of operations of the coding group with the operations of at least two of the remaining three groups. The focus is the analysis of machine-readable text material only: if a program provides multimedia functionality, supporting working with pictures and sound, this feature is reported but not reviewed. A further selection parameter concerns the operating system on which a package runs - given that the majority of researchers nowadays work with Windows, Unix or Macintosh, only programs for these platforms were selected.

One concern related to this decision was whether by not including some older but widely-used programs running on DOS, such as TACT (http://www.epas.utoronto.ca:8080/cch/TACT/tact0.html), we run the risk of missing important kinds of support or operations which should be considered. In particular, the question which arises is whether there is functionality in TACT or MECA (Carley/Palmquist 1992) that is either not available by or better than what is available in the Windows or Mac-based programs. For that reason we also tested the DOS-programs Ethnograph (http://www.QualisResearch.com), INTEXT (Klein 1997b), MECA, PLCA (Roberts 1989) and TACT. This testing has played a role in shaping the group of ‘primary’ operations discussed above.

The following software selected for the review are: AQUAD, ATLAS.ti, CoAn, Code-A-Text, DICTION, DIMAP-MCCA, HyperRESEARCH, KEDS, NUD*IST, QED, TATOE, TEXTPACK, TextSmart, WinMAXpro, and WordStat. Note that one of the authors of this review is also the developer of TATOE and the other the developer of TEXTPACK. The following programs are typically categorised as 'qualitative': AQUAD, ATLAS.ti, HyperRESEARCH, NUD*IST, QED and WinMAXpro, whereas CoAn, DICTION, DIMAP-MCCA, KEDS, TEXTPACK, TextSmart and WordStat are 'quantitative' ones. Code-A-Text and TATOE support operations which belong to both qualitative or quantitative. One should, however, be careful with this broad categorisation: the selection of the appropriate software should depend on the research question at hand and not necessarily on the above dichotomous categorisation.

4. Concrete selection process

Given the selection parameters discussed above, we first compiled a provisional list of all text analysis programs we knew or had heard of. Along with the relevant literature, e.g. Tesch (1990), Weitzman/Miles (1995), Evans (1996), Klein (1997a), we had additional sources for gathering information. These were, on the one hand, Web sites providing links to and information about text and content analysis software and, on the other hand, email communications of relevant user groups, e.g. on qualitative analysis, statistics, humanities computing. This resulted in a collection of diverse information on text analysis software, ranging from product flyers to research papers and user's manuals for some of the programs. Those text analysis programs which do not support a combination of functionality as explained in the previous section were excluded. The outcome was a shorter list of software, which was also more suitable for the purposes of the review.

The material gathered for each selected program provided us with a good idea of the background and motivations for developing some packages, the strengths of each software according to its developer(s) or distributor(s) as well as the types of users targeted and the kinds of text analysis projects each software aims at supporting.

We then examined a subset of the selected software in order to 'validate' the list of primary operations we had initially specified. More specifically, in addition to the DOS-based programs mentioned in the previous section, we reviewed the demo versions of ATLAS.ti, NUD*IST, and WordStat, the beta version of DIMAP with the MCCA content analysis module, and the full version of TextSmart. The result of both information processing and software examination was to slightly complement as well as differentiate (where appropriate) the operations we initially specified.

Software selection was not independent from cost considerations: where there was a demo or a free beta version of a program available, we used this instead of the full version. The functionality spectrum of demo versions is typically the same with that of the full versions of a selected software with the difference that the demo version may either

- be used only for a limited period of time or
- the open, import or save operations for one's own texts or the export of coding are not allowed.

The consequence of reviewing demos or beta test versions is that, unfortunately, aspects concerning the stability of the program when used intensively or the speed of processing could not be tested for all programs. We certainly consider these to be decisive factors when choosing a program. Nevertheless, we decided to concentrate instead on the examination of how and to what extent the different software packages provide for a combination of functionality for text analysis.

Since there was a deadline set for the review, only programs were considered which we had heard about or whose Windows versions were available by August 1998. However, if patches were made available for a selected program after this deadline, we also considered the updates and corrections for the review.

5. How the review is organised

The selected programs are presented in alphabetic order in the second part of the book. Each review forms a separate chapter and is organised according to the following topics:

1. A short description of the program: information about the main 'idea' and functionality of a program, the kinds of texts it can analyse, and whether a demo or test or full version of the program was used.
2. *Terminology*: a glossary with a list of terms specific to each program. The terms are alphabetically listed. All terms are grouped together in a Glossary at the end of this book.
3. *Text importing and structure*: the data entry mechanism of each program and the text structure it can handle.
4. *User interface*: general design of the program's desktop, ways of displaying text, and coding, interactivity, visualisation, etc.
5. *Text coding and categorisation*: creation and maintenance of a categorisation scheme or a dictionary, modes of coding, units of coding, memos, support for teamwork, etc.
6. *Text and coding exploration*: search and retrieval functionality for texts, text strings, codes and memos, frequency of occurrence information, KWIC displays, hyperlinking, etc.

7. *Data export and interoperability*: the program's functionality concerning the formats into which text and coding can be exported and possible interoperability with other programs.
8. *Discussion*: we selected specific features of the program and discuss them briefly.
9. *References and further information*: bibliography references, e.g. user manual, reviews, application of the program, and information about its availability.

The third part of the book presents an extensive discussion of the reviewed software. Differences and commonalities between the programs as well as the lacking features and aspects related to future development are discussed. In particular, this part is organised according to the following topics:

1. text import mechanisms and the kinds of text structure software can handle,
2. the ways software support categorisation and coding,
3. data export mechanisms,
4. kinds of linguistic processing and information used by software to enhance analysis,
5. ways of supporting the exploration of text and coding,
6. issues related to user interface design,
7. aspects concerning whether and how different modes of analysis are integrated, and
8. aspects related to the generality of use of software.

Finally, all the programs with the distributors addresses and cost information are listed in the Appendix.

PART 2: REVIEWING THE TERRITORY

1. AQUAD (Analysis of QUAlitative Data)

AQUAD 5 for Windows is a qualitative text analysis program which was developed at the University of Tuebingen, Germany by Guenter Huber. It is distributed by the publisher Ingeborg Huber Verlag, in Schwangau, Germany.

Interview transcriptions, field notes, observation protocols, diaries, historical or political documents are some of the kinds of texts that may analysed with AQUAD. It supports the construction of categories and the on-line coding of one line or more of each text based on these categories. It is also applicable for memo writing, retrieval of words or coded lines as well as exploring coding structures.

For the review, we used the 30-days demo version 5.3 of AQUAD. During the writing of this review a new improved version 5.5 was made available. We then corrected and extended, where appropriate, our review to consider the new functionality of AQUAD.

1.1 Terminology

Implicant: a module that allows the user to perform Boolean qualitative comparisons of the data (to support investigation of causality).
Linkages: search algorithms for viewing how two or more codes 'link' or co-occur in the same document.
Master Code list: a list of all codes used in all files of a project.
Meta-codes: are super-ordinate codes.

1.2 Text importing and structure

The AQUAD user creates a 'project' which includes the text data to be analysed. ASCII text files which follow the naming conventions prescribed by the program (that is the last three characters of the filename must be numbers), may be imported in AQUAD as the data files of a particular project. Pre-processing of the text files may be necessary since the texts must contain hard carriage returns and each text line may not contain more than 60 characters. A line is the single structural unit of a text in AQUAD.

Following the "basic rule of qualitative analysis that text documents should not be changed during analysis" (Huber 1997: 48), a text may not be edited once imported in

AQUAD. It can, of course, be processed and then re-imported. However, this has consequences for texts which have already been coded. Coding may not be valid for the new text data because coding in AQUAD stores line numbers together with the codes and because the line numbering for the re-imported file may be different.

1.3 User interface

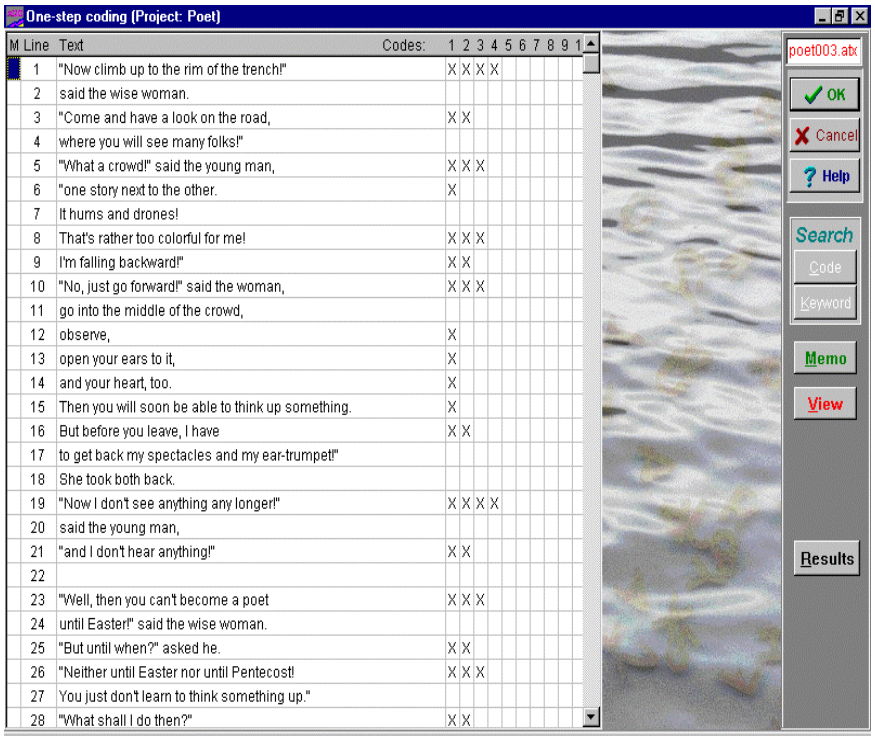
The AQUAD desktop comprises a menu bar with a number of menus for file processing, coding, and searching. The imported text is displayed only by means of the View option of the AQUAD Coding window. This window *must* be closed in order to make use of the other, non-coding features of AQUAD.

As shown in figure 1 the Coding window organises the different kinds of information in columns. The first column shows whether there is a memo for the displayed line. The second contains the line number followed by the text (of the specific line) and a table of 10 columns, each filled with a cross if a code has been assigned to the specific text line. Clicking on the cross displays the name of the code.

Each separate column can be selected. By clicking on the memo column a Memo Manager appears with either an already existing memo or empty fields for creating a new memo. Clicking the line number column brings up a window with a list of all created codes: any of these may then be selected to code a line. Clicking on the text column brings up another window with all the codes which have already been assigned to the specific line of text. This display of coding also provides information about the exact line numbers coded, i.e. from which and until which line number a code has been assigned. Finally, clicking on a code displayed in the code column highlights the coded lines and the name of the code is displayed.

All text lines together with their respective coded text segments may be displayed by using either the View menu of the AQUAD menu bar or the View option of the Coding window. This type of display shows each text line which has already been coded marked with a '+' character. Upon double-click, the respective coding is displayed underneath the line together with the lines which the coding concerns. The user may view all coding when the Expand option is chosen. Unfortunately, there is no 'De-expand' or 'Compress' option to return to the original display. Viewing the coding is cumbersome and inflexible. The edges of the text and code area are not adjustable, and although there appears to be much space in the background window, the user must always use the arrows and scroll bars to move around.

Figure 1



New codes can be created after the user has opened the Coding window displaying the text to be analysed. The user clicks on the line column for the line to be coded and as a result a pop-up window with all the already created codes appears. The name of the new code to be created and assigned to the line selected or to a number of lines can be typed in the appropriate field.

The search and retrieval operations in AQUAD are organised in four different menus, that is, Retrieval (search or count of codes or words, etc.), Tables, Linkages, and Implicants, or can be initiated from the Coding window. The result of searches initiated from the specific menus are 'on-screen' reports. The Keyword and Code search options available from the Coding window search the opened file of a project for the specified keyword or code. The matches to the search are displayed on a separate window together with the line number for each occurrence. Global searches over all text files of a single project cannot be de-

fined and initiated from the Coding window; instead the user must close the Coding window and use the Retrieval menu options.

AQUAD provides on-line help which is organised as a topic index. The user can also search the index by typing in a keyword.

1.4 Text coding and categorisation

The user may create any number of codes which are assigned to text lines. The codes have short names. On the basis of codes, Meta-codes may be defined: these are 'super' or broader codes. In this manner, AQUAD supports the construction of a hierarchical coding scheme. AQUAD creates a Master Code List which comprises all codes used in all files of a project.

The AQUAD user may code either a line or a number of lines; no other unit, such as a word or a phrase, may be coded. Two 'modes' of coding are supported, namely one-step and two-step coding. The former means the on-line coding, whereas the latter is relevant for those users whose texts are not in machine-readable form and who still wish to use AQUAD. The two-step coding process is as following: First, start with a print-out of the text files, line-numbered by a word processor. Second, mark the beginnings and ends of relevant text portions, and write a code in the margin. Third, enter the coding information into AQUAD, using the line numbers and the code names.

AQUAD supports nested coding, where parts of an already coded text segment, i.e. a line or a number of lines, can be assigned to different codes. Furthermore, a coded text segment may be further assigned to other codes.

The maintenance of the coding lists is supported in AQUAD by enabling the user to delete codes either locally or globally (removal of a code from only one file of a project or from all files of a project), rename codes or obtain a list of used or non-used codes.

Memo writing is supported by AQUAD. The user may relate memos to a text, to a line number, to a code and to an additional index. Furthermore, text search in memos is supported.

1.5 Text and coding exploration

The user may search for words or codes or word suffixes over all text files of a project either based on a 'catalogue' containing a list of words or by directly typing in the word(s) or code(s). The search results are displayed on a separate window with each word or code together with its respective frequency of occurrence. The Suffix search option in AQUAD is in effect a word search, but without treating the word as a single token. AQUAD will

not generate a list of all words of a project which the user can use for search and retrieve operations.

In addition, a search may be performed combining codes and keywords. The user specifies both one or more codes and one or more keywords to be searched; AQUAD then searches the project for all occurrences of the specified keywords within the lines assigned to the specified codes and displays the result in a report.

An attractive search feature in AQUAD concerns the possibility to explore the coding structures. A variety of options are available for this purpose, namely, the user can search for

- *nested codes*: coded line(s) which contain another coded segment within its first and last line of coding;
- *overlapping codes*: all codes which were attached to overlapping text segments;
- *multiple codes*: all coded lines which were coded with more than one code;
- *redundant structures*: a heuristic approach supporting the detection of associations. AQUAD will retrieve those instances where a code appears more than once within a specified number of lines for one or more codes specified
- *sequence of codes*: which codes appear in the neighbourhood of specified codes (nested and overlapping codes will be included).

The user may search for a word or a phrase, and by using the AQUAD Keyword option of the Coding window can see the retrieved instance(s) in text. The search is case sensitive but does not treat a word as a single token: e.g. searching for all occurrences of 'you', will retrieve not only all instances of the word 'you', but also of 'your' and 'yours'. To avoid this the user must include blanks in the search. The matches are displayed in a separate window with each occurrence shown in a separate line with the particular line number. In a similar way, the user may search for a code or search for and replace a code. Note that there is no interlinking between the Coding window and the window with the search results. This means the user can neither leave the results window open and go on with coding nor perform any operations on it.

There is some very 'elementary' KWIC functionality offered by AQUAD. One first specifies in the main menu a word or list of words to be retrieved, and then chooses one (*not* all) of the project files. The display of the Coding window then comprises a new column with a little square icon to mark those lines where the keyword was found and the same icon is displayed in front of the word in the text line.

Tables and Linkages help to explore the existing coding and the possible relations between the codes. Tables are in principle cross tables, whereby the rows and columns of a table are specified codes, and show the association of codes. The result of a table analysis may

be either a list of all text segments fulfilling the double conditions for cells of the table, or a frequency table to show how often conceptual codes appear in the texts under the condition of profile codes, or a list containing in its cells only coded lines which fulfil the combined criteria of column and row headers. It is a three-step process to use the tables: first create a table, then save it, and then retrieve it.

Linkages check how two or more codes occur within a specified distance, i.e. in a number of lines. The distance and the codes must be provided by the user. AQUAD supports the user to define her own Linkages or to use pre-defined Linkages.

The AQUAD module Implicants transforms case-specific configurations of meaning into truth values of a binary logic (i.e., a particular characteristic is 'given/is not given'), and then combines these truth values according to the rules of Boolean algebra. Implicants in AQUAD make use of logical minimisation (see Ragin 1987) in order to compare the results of qualitative analyses. In particular it is used to compare Linkages of meaning or configurations of categories when examining a large number of single cases or when meta-analysing qualitative studies.

1.6 Data export and interoperability

AQUAD exports neither the coding scheme nor the coding results to another format for further processing. The coding is automatically saved by the system within the AQUAD project.

AQUAD will save counts of frequencies of occurrence as ASCII files. These can then be imported into statistics packages after the necessary pre-processing, e.g. removing the headers and defining the variables.

1.7 Discussion

AQUAD has been designed to assist qualitative text analysis and theory building and it may support this to a certain extent. It is not as user-friendly or flexible as other qualitative text analysis programs, such as ATLAS.ti or WinMAXpro. However, it *does* offer some search and retrieval functionality concerning coding, such as Linkages, which can be attractive for particular types of analyses.

For those interested in coding line-by-line, AQUAD may be the program to use, since coding as well as both keyword and code retrieval in AQUAD are based on text lines. The developer of AQUAD claims that this enforces the user to edit the text in a "meaningful way", since the user herself must decide which text segments need to start with a new line because line numbers and codes come together. In effect, AQUAD demands a first interpretation and coding of text during the transcription process prior to using AQUAD.

No form of automatic coding is supported in AQUAD: the user cannot assign either all occurrences of a word or a phrase or all occurrences coded according to a combination of codes to one or more codes in one go. Any existing socio-demographic data for each project document which the user wishes to consider for the analysis cannot be imported in AQUAD.

Counts of frequencies of occurrences of codes can be saved as ASCII files. After appropriate processing, i.e. removing the headers, these can then be analysed with statistics packages. Kuckartz (1998) reports that the developers of AQUAD, INTEXT, and WinMAXpro have agreed on a 'standard' format for data exchange among them. This would allow data analysed in AQUAD to be saved in a format that the other two programs could interpret and work with. However, as far as we know, this has not yet been realised in AQUAD.

The design and functionality of the Coding window, where coding is displayed, is not particularly user-friendly. The coding display is redundant and the design of the window cannot be flexibly re-adjusted to see the columns to the far right.

Despite the type of search the user initiates, the result of a search is always displayed in the same way which makes it difficult to distinguish between searches. Moreover, the results display is not live and the user can select one or more occurrences and see them in text. In general, a main disadvantage of the user interface is that the AQUAD windows do not communicate with each other.

1.8 References and further Information

The AQUAD User's Manual (Huber 1997) is available in electronic form or may be ordered (at an extra price). The tenor of the manual is at times as if it were written to address students or novices to qualitative text analysis as well as novices in using computers for analysis. A shorter version for experienced users or a short tutorial would be useful.

Huber/García (1992) report on using (an older version of) AQUAD. Earlier as well as current versions of the software are reviewed also in Weitzman/Miles (1995), Kelle (1995), and Kelle (1996).

Those interested can obtain more information about AQUAD and recent developments from the following Web site: http://www.aquad.de.

2. ATLAS.ti – The Knowledge Workbench

ATLAS.ti is a text analysis program running under Windows.[1] It supports the researcher during the text interpretation process, particularly in managing, coding, memoing, extracting, exploring, and comparing text and codes. The codes which the user defines and assigns to text segments can be linked in networks with a varying degree of complexity.

ATLAS.ti offers powerful code retrieval functions with Boolean, semantic and proximity based operators. Furthermore, the ATLAS.ti user can generate an SPSS-readable file for further statistical analysis. The types of texts which may be analysed in ATLAS.ti vary from in-depth interviews or observation notes of secondary analysis to speeches or analysis of data from a series of focus group interviews. In general, ATLAS.ti will work with any kind of text as well as non-text material, such as graphic or audio files.

For the review of ATLAS.ti we used the demonstration (free of charge) version 4.2 with service packs 1 and 2, which is a restricted but fully functional copy of ATLAS.ti. In comparison to the full version, the demo allows only a restricted number/size of projects. The demo version contains exemplary texts with some coding as well as a file with a "Preliminary Early Bird's First Steps Crash Course Short Manual".

2.1 Terminology

Family: different object types, i.e. Primary Documents, codes, or memos can be grouped in families. One code or Primary Document or memo may belong to more than one family.
Hermeneutic Unit: the 'container' for an analysis project. Primary Documents are assigned to a Hermeneutic Unit, which is the most prominent data structure the user is working with.
Primary Documents: the original documents to be analysed and coded.
Quotations: each text segment (a word, a group of words, a whole paragraph, etc.) which is assigned to a code forms a quotation.
Search Categories or *Search Swarms* or S*earch Patterns*: definitions of search strings which may be given a name. Another term used for this purpose is a definition of a text search.
Super Code: different from normal codes in that it stores a query formed using other codes.

2.2 Text importing and structure

To start working with ATLAS.ti the user creates a Hermeneutic Unit. The raw data can be text, graphic, or audio data. Any of this type can be assigned to a Hermeneutic Unit as a

---

1 The development of ATLAS.ti started in 1989 at the Technical University of Berlin. Starting in 1993, the prototype was further developed to its first commercial release by Thomas Muhr.

Primary Document. Text data must be ASCII/ANSI files, audio data must be WAV data, and a number of formats for photos or maps, e.g. JPEG, TIF, BMP, etc., is supported. The Primary Documents which are assigned to a Hermeneutic Unit, i.e. the main data structure of ATLAS.ti, do not become part of it.

In addition to assigning texts or non-text material to a Hermeneutic Unit, the user can import and assign to the unit a file in ASCII/ANSI format with a list of categories (codes) for coding the texts.

In order for ATLAS.ti to efficiently process the text contained in Primary Documents, the text must be in an ATLAS.ti specific form. The user must ensure that hard lines are inserted in the document, since ATLAS.ti does not use automatic line-wrapping. The user must use the hard-line-feed option while preparing the documents, otherwise every paragraph will be seen as one - possibly long - line.

Every text unit which may be a candidate for coding, for example, a dialogue turn or an answer to a survey, may be processed as a separate paragraph. ATLAS.ti recognises paragraphs only if paragraph breaks, i.e. two empty lines, have been used. Hence, those text files where paragraphs are differently delimited must be pre-processed.

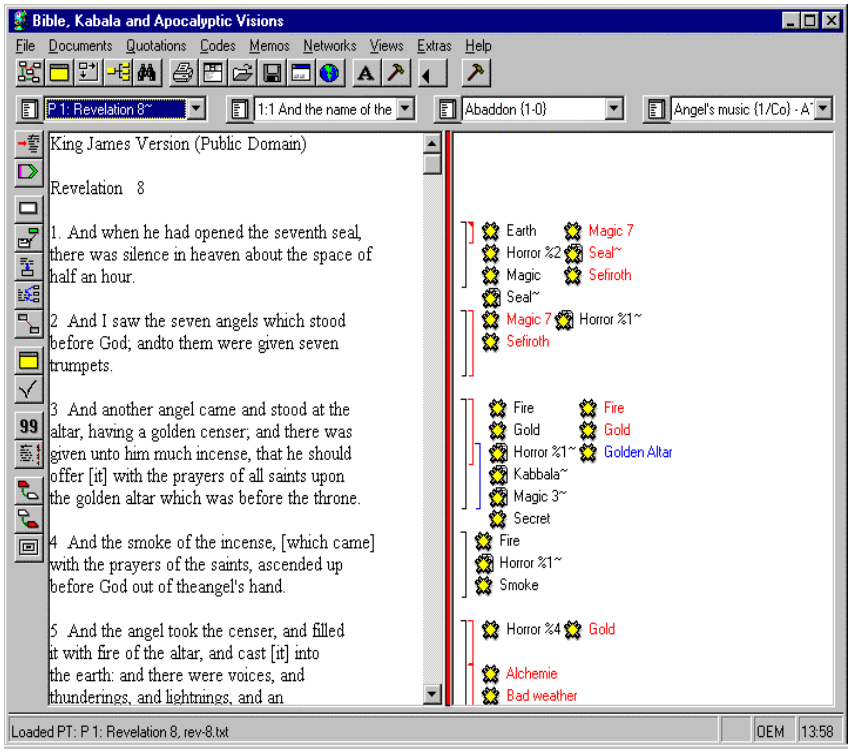
Primary Documents are read-only documents; they are viewed as part of an archive, as frozen and shared documents. Nevertheless, the Primary Documents pane, i.e. the main window showing the text, can be switched from a read-only to an edit mode, the only restriction being that ATLAS.ti will allow only the user logged as 'administrator' to edit the Primary Documents. Editing the Primary Documents which have already been assigned to codes may, however, result in corrupting the in sync functionality of the ATLAS.ti user interface as far as the coded segments and the text positions of these segments are concerned. In other words, the result of adding or removing text of a Primary Document may be that the integrity of 'hyper-linking' between coded positions, i.e. quotations, and documents will be corrupted, so that if the user in a later stage activates a coded text segment ATLAS.ti may highlight wrong passages in the document. Despite the above 'disadvantage' it is worthwhile mentioning that there are advantages in assuming that the documents analysed are read-only. This is true in particular for the overlay technique of ATLAS.ti, for instance, the possibility of multiple coding, the sharing of the same documents among users, the direct processing of read-only documents e.g. CD-ROM texts, etc.

2.3 User interface

ATLAS.ti makes use of object-oriented user interface design which tries to keep the necessary operations close to the data they are applied to. The main window of the main pane of the program contains the Hermeneutic Unit Editor, which displays one of the Primary

Documents as selected by the user. This way the user has always in front of her the text data she is working with. Note that only one Primary Document may selected and displayed at a time.

Figure 2



Apart from the menus, the ATLAS.ti desktop makes use of two icon bars: the main Tool Bar (horizontal) including functions which are also available as menu options, and the Primary Document Tool Bar (vertical) which contains options related either to the display of text and coding, e.g. show line number or show quotations, or to coding operations, e.g. open coding or create memo. The Hermeneutic Unit Editor of ATLAS.ti is shown in figure 2.

ATLAS.ti makes extensive use of Windows technology: it contains drop-down lists, which are advantageous for listing potentially large numbers of objects, such as coded text segments, in little space. In particular, ATLAS.ti offers one drop-down list for selecting

between the Primary Documents in a Hermeneutic Unit, one for selecting a code, one for memos, and one for quotations. Only one item may be selected from a drop-down list at a time. Multiple selection is possible from object lists, that is, lists of the different object types (Primary Documents, codes, memos or coded instances). One or more items of the object list can be selected and a number further operations, e.g. network views, can be performed.

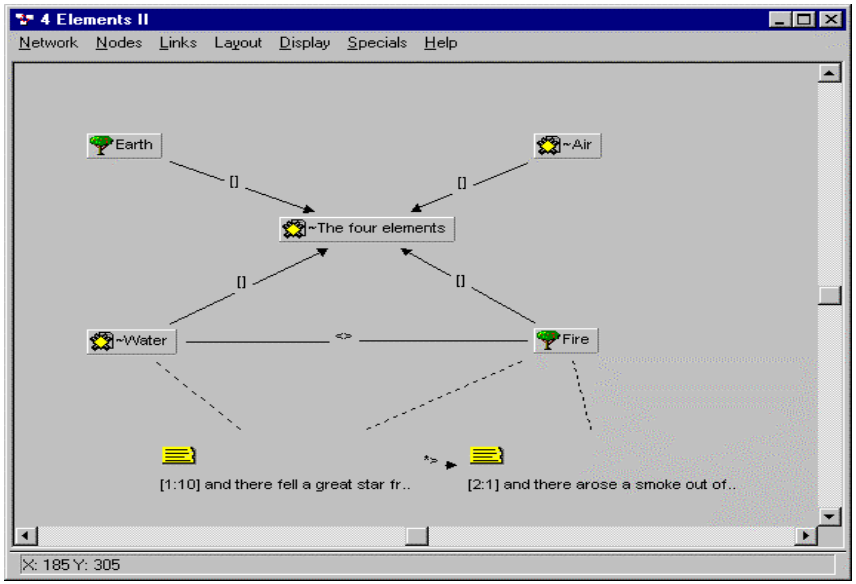
ATLAS.ti makes also use of context-sensitive pop-up menus for commands applying to certain objects or panes. Pop-up menus are dependent on the type and current state of the object or context in which the menu was activated.

In addition to the Hermeneutic Unit Editor, the program provides a number of different windows and editors: the Network Editor, the QueryTool and the Object Explorer. The former displays codes, quotations, and memos as well as Primary Documents as nodes of a network. The user can manipulate each of them similarly to when using a drawing tool: items may be selected and moved around in a Network Editor display and links may be drawn or cut between them. The user may specify the amount of information displayed for the quotations in a network view by selecting the relevant option from the Network Editor's Display menu. Figure 3 shows a network view displaying a number of codes and quotations.

Network views may be saved under a name, stored and accessed inside the Hermeneutic Unit they have been created, and they may be commented.

Two methods of automatic layout are possible in network views, namely semantic and topological. The former method uses a semantic layout algorithm which "tries to place the nodes into optimal positions using an invisible matrix of default positions. It tries to display the nodes with the highest connectivity into center positions. By recursively applying the same method to the rest of the nodes using neighboring positions of the first node placed, the algorithm tries to avoid overlapping nodes and too many crossing links. However an optimal solution cannot be found in a reasonable time." (Muhr 1997: 74). Results of this procedure which are not as optimal as the user would wish for can nevertheless be refined manually. The topological layout method creates a sequential list of nodes which is positioned diagonally from the upper left to the lower right of the Editor window.

Figure 3



The QueryTool window is the richest means the ATLAS.ti user has for exploring coding and hypothesis testing. The Boolean, semantic, and proximity operators available are listed as click-on icons. The window is organised in different panes for displaying the codes as well as the Family Codes, the query being constructed and the result list, which displays the retrieved coded segments according to a constructed query. The hits displayed are in sync with the ATLAS.ti main text pane and clicking on a hit causes it to be displayed on the text pane.

The Object Explorer window resembles the Windows 95 and Windows NT Explorer and its purpose is to provide a structured display or a kind of a table of contents of all the object types of a Hermeneutic Unit. It lists all Primary Document, codes, memos, families and network views. The quotations are listed under the Primary Documents. The user can navigate around the hierarchically organised object types.

The on-line help of ATLAS.ti is organised both according to contents and index of keywords which can be searched. In general, the Help menu option of ATLAS.ti provides detailed information about the various components and features of the program.

2.4 Text coding and categorisation

The assignment of one or more codes to text segments in ATLAS.ti can be performed either manually or automatically. Manual coding is on-line. The user selects a text segment – in fact, she can make free-hand selections of arbitrary size with the smallest selection being a single character - and has four different options for assigning the highlighted text segment to a code:

- *Open coding*: The program creates a new code and simultaneously codes accordingly the selected text. The user uses a button or the relevant menu option for this kind of coding and types in the name of the code to be assigned. The list of all codes is updated with the new code entered in the list.
- *In-vivo coding*: If the selected text segment itself offers a good name for the code, then the user can use this coding technique to code and at the same time create a code with the same name as the quotation just created. Of course, this coding technique is possible only for text material.
- *Code-by-list*: This technique may be used when assigning existing codes. Any number of codes from a multiple-choice list on a separate window can be chosen or the user can use drag-and-drop to code a text segment. If the selection equals an existing quotation only those codes which have not already been assigned to the quotation are offered in the list.
- *Code-by-list inverse*: The standard code-by-list technique associates a list of codes to one selected quotation. Once in a while, it may make more sense to associate a list of quotations to one code.

A further way to code a text segment in ATLAS.ti is by means of ‘quick coding’. This coding technique starts by selecting a code as opposed to selecting a text segment first and then the user assigns the selected code to a text segment. This is a fast and efficient method for the consecutive coding of text segments using the same code. To use quick coding the user selects a code in the Code List window and makes it the current code; she then makes a selection in the Primary Document and clicks the Quick Coding button or chooses the relevant menu option.

As far as automatic coding is concerned, this can be performed in connection with a text search (see search categories in section 2.1). Using the Auto Coding facility of ATLAS.ti, the user first types in a text search and she either provides an already created code or creates a new code according to which the resulted hit should be coded. She then specifies the scope of the search, which can be either the displayed Primary Document or all Primary Documents or a family of them, and finally specifies the size of the unit to be coded, that is whether the exact string matching the search or a word or a line or a sentence or a

paragraph or the whole text should be coded. ATLAS.ti will then either code all the relevant text segments automatically in one go or, alternatively, the user can choose to manually confirm each hit before it is coded.

ATLAS.ti supports multiple, nested, and overlapping coding and imposes no restriction on the amount of codes which may be assigned to a quotation and no restriction on the numbers of quotations assigned to a code. Code creation is, in fact, easy and straightforward in ATLAS.ti. The user can delete or rename codes with the same easiness she can create ones. In addition, codes may be merged to a new code with all their quotations or other references kept.

ATLAS.ti makes a distinction between normal codes and Super Codes: the latter are as transparent to the user as the normal codes, and in fact are included in the code list, although some operations, e.g. code-by-list, cannot be applied to them. Dissimilar to normal codes, Super Codes store a query constructed in the QueryTool of ATLAS.ti. For example, if a query has been constructed for looking for quotation coded *x and y* this can be stored as Super Code, let us call it *z*, and the retrieved quotations may be assigned to this Super Code. It is important to note that Super Codes store only the query itself and not the retrieved hits. The advantage of this is that the retrieved instances are always newly-compiled. Super Codes can be included in new queries and can be saved in further Super Codes.

The Code Families[2] functionality of ATLAS.ti helps the user to group a number of categories together, whereby one category may belong to more different code families. This way the user may organise codes and their respective coded segments in meaningful groups and, at the same time, she has the possibility to combine codes from different families for searches or coding. This is helpful both for text exploration and further coding.

The best way to think of coding in ATLAS.ti is as a kind of independent transparency layer over the text material analysed. The text displayed is not changed, edited or modified, but rather enriched with an additional layer of information which is *linked to* rather than *included* in it.

The user may create a network of concepts, whereby different relations are available for connecting the concepts with each other. ATLAS.ti offers six default relations, also called Named Links: is_part_of, is_associated_with, is_cause_of, is_a, is_property_of and contradicts. These relations are semantic and can be used for linking two codes or two quota-

---

2 Note that not only code, but also memo and Primary Document families may be created in ATLAS.ti.

tions with each other. The links between quotations and codes are 'hard-wired' untyped relations.

In addition to the available semantic relations, the program supports the user in defining new relations if the ones offered do not cover the user's needs. User-defined relations are possible only for code-code or quotation-quotation links. The user is guided through different options in order to define a new relation so that the system can make appropriate use of it in the subsequent processing. A prerequisite for this process is however, that the user is familiar with the meaning of the options offered; for example, the user must know beforehand what a symmetric, an asymmetric, and a transitive relation is.

Two methods are available for creating network views: the first one creates an empty view, i.e. tabula rasa, whereas the second one creates a network view from a selected object and its direct neighbours, i.e. focused network view. The latter is created on-the fly from the selected object.

ATLAS.ti supports memo writing and memo management. Memos may be attached to a quotation, a code or a code family as well as to a Primary Document or Document family, a network view, a code-to-code link or quotation-to-quotation link or to a Hermeneutic Unit. Memos can be sorted, organised in families, filtered, and edited in ATLAS.ti.

Coding is displayed in ATLAS.ti in the so-called Margin Area and its display is interactive: the user can select a code on the Margin Area and the respective coded segment is immediately highlighted in the Primary Document as a response to the selection. Note also that the codes at the Margin Area can be activated and that in this way a number of operations (e.g. list all quotations of the selected code or add a comment to the code) are possible by means of existing context-sensitive menus.

The manner in which the existing coding is displayed can be rather confusing. ATLAS.ti makes use of visual graphics, colour, icons, but these are displayed at the right side and not *on* text. In the case of fairly rich coding apart from the use of the same colour for underlining the coded text segment and displaying the code to which this is assigned, it is difficult to quickly recognise which part belongs to which code unless one highlights each and every code.

ATLAS.ti supports teamwork by allowing multi-authoring, although this is not simultaneous. Every object type, that is, Hermeneutic Unit, quotation, code, memo, etc., is automatically assigned a 'stamp' for date and author. User management defines the administrator, the 'authorised' authors or co-authors, or the group of users which may access or modify Hermeneutic Units. The Merge Tool functionality of ATLAS.ti is another way

for supporting different researchers using the same tool and working for the same project; its role is to merge the contributions of different users sharing a common coding scheme.

2.5 Text and coding exploration

Four different types of objects may be searched for in ATLAS.ti, namely, the Primary Documents, the codes, the memos, and the quotations.

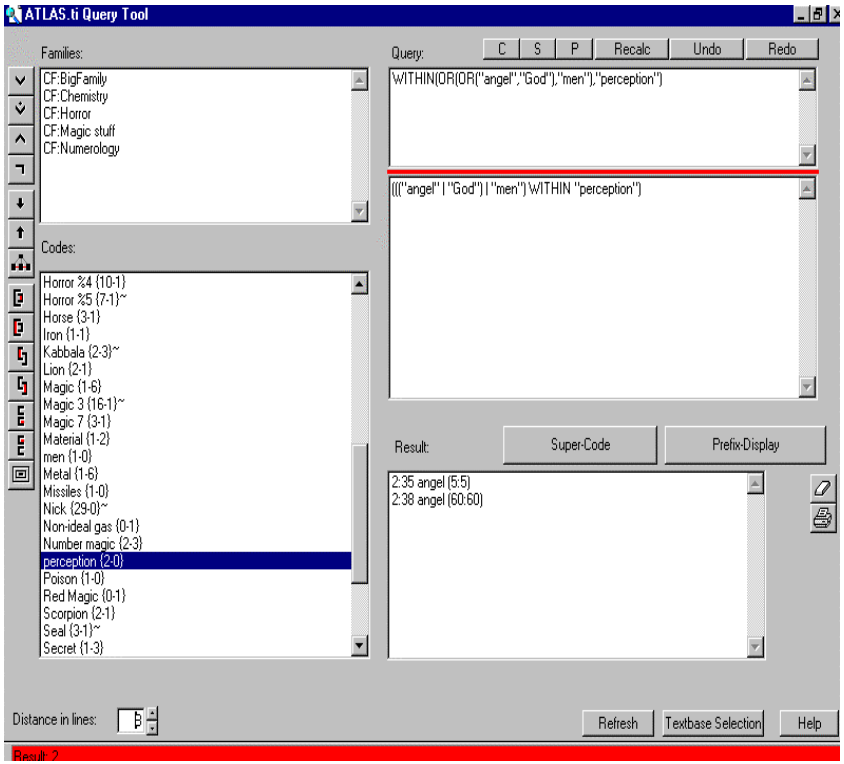
There are three kinds of text and code search offered: (i) string search, (ii) pattern search and (iii) GREP search. In terms of the former, if a text string is typed in, ATLAS.ti starts searching for it and highlights the first occurrence it finds. The search, however, does not take into account word boundaries so that for a text string such as 'the' a text segment fa'the'r, which contains the string 'the', will also be retrieved! A way to avoid this is to opt for a pattern search instead of a string search and add the character '|' after the word 'the'.

The user can search not only for strings, but also for search swarms or (parallel) 'patterns' or 'categories'. These are defined as search strings which are given a name. Search categories can be stored under descriptive names, they can build hierarchies and are managed in search libraries in ATLAS.ti. These categories can be used for further searches, mixing text strings and category searches together. However, the only possible operator between the parts of a search category is OR: for example, a search category may be defined which says 'search for the string *the* or the category *CAUSE* or the string *to*'. It is not possible to define a search category that says 'search for the word or text string *the* followed directly by any string assigned to the code *CAUSE*'. In that sense there are still considerable limitations for this search mode. The OR operator used for the definition of search swarms is mainly a means for searching for groups of synonyms.

The GREP search enables searching using regular expressions. In particular the following nine regular expressions may be used for defining GREP searches: *, $, ., +,[], :d, ?, \ and ^.

A very powerful and advanced feature of ATLAS.ti for searches is the QueryTool. This may be used for searching the existing coding, i.e. not the text of the Primary Documents, but rather the existing quotations. Boolean operators, e.g. OR, XOR, AND, NOT, as well as semantic operators, e.g. SUB, UP and Siblings, and proximity operators may be used for searching the coded text segments. For example, the query desplayed in Figure 4 searches for all text segments assigned to either the category 'angel' or 'God' or 'men' which occur within the coded segments assigned to the category 'perception'. The semantic operators search for all coded text segments according to a code taking into account the code's broader, narrower or sibling codes. The user may specify the number of lines for the maximum distance between two coded text segments for each query.

Figure 4



The QueryTool is very powerful, but may give a fairly complicated first impression. However, with this tool the user has at her disposal a flexible medium to *really* explore existing coding of a project.

ATLAS.ti uses the principle of Reversed Polish Notation (RPN) as its query language for building queries in the QueryTool. Although one may argue about whether this is too complicated for text analysis or qualitative researchers, one has to admit that it allows for a varied and rich search.

Although ATLAS.ti comes with advanced search functionality such as that offered by the QueryTool, the presentation of the results of a search is rather poor. In fact, the only feed-

back provided is the relevant text position highlighted in the text every time a hit is made. The only way for the user to view the next hit is to click on the available buttons for 'next' or 'previous'. Although the search goes over all textual Primary Documents, there is no 'global' presentation of results. Note that although the search pattern can be both stored and be used to build hierarchies the results of a search according to a search pattern cannot. If the user wishes to view the results of search already performed, she has to initiate the search again.

The WordCrunch operation in ATLAS.ti generates a list of all words occurring in a single (selected) text of a project, together with each word's frequency of occurrence and the type/token ratio for this document. The user may specify a stop word list as an external file which may contain not only full words but also words with wildcards (e.g. *ask**). ATLAS.ti will not consider these words when counting word frequencies and will not include them in the generated word list. The WordCrunch output is not interactive as other displays in ATLAS.ti are, but it can be saved in a file, or, alternatively, it may be sent directly to printer or be viewed in an editor.

An important feature of ATLAS.ti for exploration purposes concerns its 'hypertext functionality'. The ATLAS.ti code-and-retrieve support involves different ways of linking texts, categories, coded segments, memos and graphs of hypertext-like networks with each other. These linking mechanisms are means for navigating from one type of link to the next. They assist in exploring both the text and its coding, and are an important means for the exploration of argumentation in text, storylines, etc.

2.6 Export formats and interoperability

The whole collection of ATLAS.ti objects, that is texts, codes, memos, etc., can be saved as ASCII files. Code networks may be exported in a special format for further use in a new Hermeneutic Unit or they can be used in other applications, e.g. Word or Corel.

It is possible to export the coding from ATLAS.ti into the SPSS statistics software. The program generates an SPSS syntax file, which can be directly used with SPSS. The information contained in the file is that for every code which has been used an one (1) is assigned; otherwise, a zero (0) is assigned if the code has not been used. The text unit over which this is calculated is a paragraph. All categories are shown with labels. Additionally, the exact positions of the quotations in a paragraph are also provided, although it is not clear how this is useful for subsequent statistical processing with SPSS.

ATLAS.ti can generate reports in HTML format which contain a list of links of the different information types of the Hermeneutic Unit (e.g. Primary Documents, codes and coded

segments, saved network views, memos, code families, etc.) which the user can use for publishing the project results electronically on the Web.

Recent (October 1998) additions to the export functionality of ATLAS.ti concern the drag-and-drop possibility of Word (or other OLE-2 Drag & Drop Servers) documents into ATLAS.ti and the export of memos as XML (eXtensible Markup Language) encoded files.

2.7 Discussion

ATLAS.ti is an advanced program for qualitative text analysis with rich functionality for data management, code-and-retrieve operations, manual building of concept networks, multi-authoring and export of analysis data for statistical processing or presentation in the Web. We consider ATLAS.ti to be one of the most interesting and all-round qualitative text analysis software available at the moment.

Maximal flexibility for the user as far as code creation and manipulation are concerned seems to be the main feature of ATLAS.ti. Indeed the user of ATLAS.ti has the possibility to build and expand a coding system, relate codes in various ways, assign them to different families, keep memos on codes, text, and coded segments with great flexibility. This advantage of ATLAS.ti however, may be a disadvantage if the user is not disciplined enough and may make it difficult to maintain an overview over the analysis performed and the code scheme built.

Because of the richness of its functionality, learning how to use ATLAS.ti may be demanding, but for those with a serious interest in text analysis and with fairly complex projects it may be worth the effort. Two of the most distinguishing features of the program in comparison with the other programs reviewed here are its Network Editor and the different ways available for building a coding scheme, allowing one to distinguish between different kinds of relations. In fact, we doubt that other text analysis systems can compete with ATLAS.ti on this aspect.

It is possible in ATLAS.ti to import a list of categories to be used for coding as well as to export the coding scheme created as an ASCII/ANSI file. In this way codes which have been used for analysing another Hermeneutic Unit can be imported in ATLAS.ti and reused. As noted in the section on text importing and structure above, every text unit which may be a candidate for coding, e.g. a dialogue turn or an answer to a survey, may be processed as a separate paragraph. This might be problematic if the user wants to identify paragraphs according to different functions, say a question and an answer in an interview, comments about the reactions of interviewees, etc. If the texts to be analysed are already delimited or encoded containing such information, this will be lost in ATLAS.ti. One way round this limitation would be to code in ATLAS.ti the relevant paragraphs accordingly.

But the restriction that ATLAS.ti cannot deal with already structured text remains in that it neither recognises nor can make meaningful use of encoded text.

An advantageous feature of ATLAS.ti for text exploration purposes is that search categories (otherwise called Search Swarms) can be stored under descriptive names and managed in search libraries. These categories may be included in definitions of new Search Swarms with the possibility to mix text strings and Search Swarm names together. A feature which we believe would improve this functionality concerns the possibility to keep a library - not only of the definitions of search swarms, but also of retrieved hits of a search already performed, making them thus available for later re-inspection. Having to repeat the search every time one wants to view the retrieved results of a search swarm can be fairly tedious.

The display of the results of a search for a text segment (a word or a string of words) or all coded segments for a given code does not help the user to obtain an overall picture of how what is searched for is used in one document or in the whole collection of the documents assigned to a Hermeneutic Unit. A KWIC view for the results of a search, perhaps coupled with an inter-linking between each retrieved instance and its position in the source text, that is in the Primary Document, would improve the exploration and inspection functionality of ATLAS.ti.

The Family Codes functionality supports the user to group different codes together, thus possibly enabling different kinds of coding to be kept together, e.g. interpretative, structural or explorative. Organising codes in larger and separate groups, and at the same time being able to combine these groups for both searches and further coding has benefits for both text exploration and further coding.

Finally, the display of coding on the Margin Area may work if the amount of coding is small, but with a dense coding it may become unreadable. This however, is a general problem of the appropriate way to display codes on text. The network display can become cluttered, and, hence, difficult to read. Interesting would be to see how automatic visualisation techniques could be used to find a way round this problem.

2.8 References and further information

The demonstration version is accompanied with a short introduction to the basics of using ATLAS.ti and provides a fairly good overview of the program for the new user, but, as the software is fairly rich, one has to be aware that what is presented in the manual is only a start. The full user version of ATLAS.ti comes with a manual (in PDF format) of circa 300 pages with plenty of diagrams, screen dumps, and examples. In addition, the full version includes a Short-Manual of circa 100 pages.

The complete manual can be downloaded from the ATLAS.ti Web site: http://www.atlasti.de. The site provides plenty of information about projects using ATLAS.ti, references, updates, and institutes or corporations using the software.

A users mailing list is also set up for information exchange and peer-to-peer support ATLAS.ti. The program is distributed by Scolari, Sage Publications, see http://www.scolari.co.uk/atlasti/atlasti.html.

For a review of earlier as well as current versions of the software see also Weitzman/Miles (1995), Kelle (1995), Kelle (1996), Schmittbetz (1997), Struebing (1997) and Walsh/Lavalli (1997).

Some hints for working with ATLAS.ti are given in Muhr (1996). A more detailed literature reference is available on the ATLAS.ti Web site.

3. CoAn (Content Analysis)

CoAn is a Windows 3.1 program for computer-assisted content analysis.[3] Based on a dictionary file, CoAn codes texts automatically as well as interactively. The program provides frequency of occurrence word lists, KWIC displays for all the coded words of a text, word comparison lists for two different texts and allows the user to export the coding to an SPSS readable file.

CoAn appears to be targeted only to German users since the short Tutorial, the Help texts and the program menu options are all in German.

The program is not tailored for a specific text type, but rather different kinds of texts can be analysed. For the review we have used the CoAn beta version 2.0, which is also the demo version.

3.1 Terminology

Category System: Dictionary on the basis of which automatic coding can happen. It contains the category numbers, and the search expressions and may contain parameters (e.g. whether upper-lower case should be considered or not).
Search Expressions: Words, stems or strings of words contained in the Category System. A specific syntax is used for differentiating between these.

3.2 Text importing and structure

CoAn accepts a raw ASCII text file. This file may contain identifiers (IDs) denoting text units as input and creates a system file. During this process, the program analyses the identifiers by means of which the structural units of a text are represented e.g. $,-, discards blanks, returns, etc., and separates words from punctuation by inserting spaces between words and punctuation marks.

Three 'levels' for IDs may be differentiated, whereby every level represents some external feature of the text. The ID values denote the features which are coded. For example, answers to open-ended questions may be structured according to three levels: the first level may be the ID of the person questioned, the second level may be the question ID, that is the question to which the text analysed belongs to, and the third level may be a sentence ID.

The user may edit the text as well as the analysis results (e.g. KWIC displays, word index) while using CoAn.

---

3 It has been developed by Matthias Romppel at the University of Goettingen, Germany.

3.3 User interface

The CoAn working environment is simply designed and organised. All operations are controlled by appropriate selections at the menu bar. The program reads a file, processes it according to the menu selection, and writes the result to a file which the user can open either from the File menu or the relevant option in the Project Files menu. CoAn is menu-driven with no possibility for on-screen selection. The user should not expect to find sophisticated GUI technology, such as visualisation, drag-and-drop, pop-up menus, but rather a 'bread-and-butter' working environment containing a number of operations typical of computer-assisted content analysis and all organised in menus.

The analysis process requires three different steps which means three different menus: Settings to define the options for the next procedure, Analysis to run a procedure, and Project files to see the result of an analysis procedure.

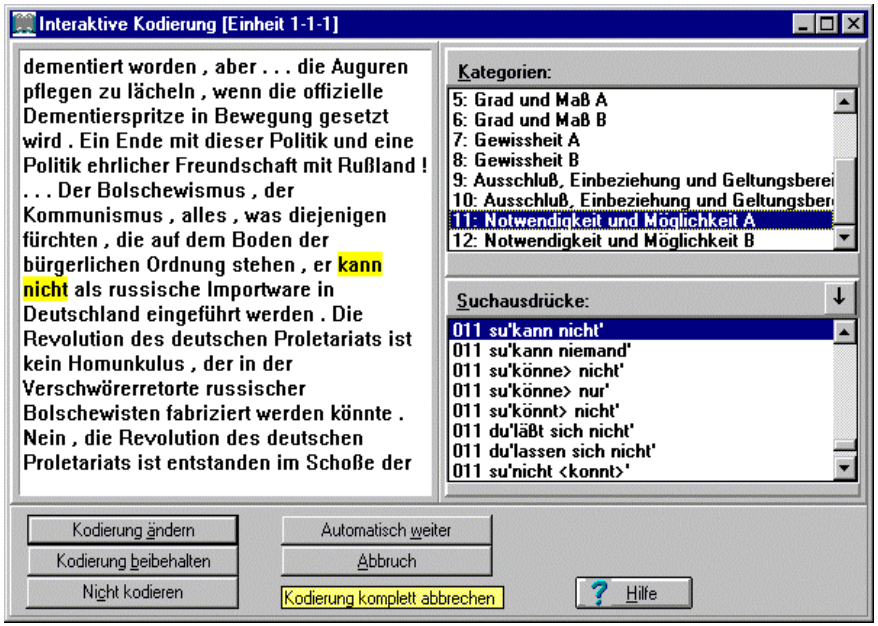
3.4 Text coding and categorisation

Coding in CoAn can be performed either automatically or in an interactive manner and is based on Category System, i.e. a dictionary. The dictionary entries can be both single words as well as strings of words. The latter can be 'strict word sequences', that is one word followed exactly by another word, or, alternatively, they can be 'loose sequences', i.e. one word followed by one or more words in the same unit and the specified sequence, or all words of the string appearing in the same unit (the sequence doesn't matter). Words or word sequences may contain wild card characters and may be defined either case sensitive or not case sensitive. The dictionary entries are organised in categories and each category is identified by a number and a label.

New categories may be calculated on the basis of the frequency of occurrence of the words or categories in a unit (e.g. sum of specified categories, relative frequencies).

CoAn codes automatically a text on the basis of a dictionary specified by the user. Furthermore, the user can 'mark' a number of dictionary entries for interactive coding. In that case whenever CoAn finds an occurrence of a marked word it asks the user if it should code it or not as shown in the example in figure 3: CoAn asks whether the highlighted string 'kann nicht' should be assigned to the category 11. If the answer is negative it then asks whether another code should be assigned. Note that CoAn does not support multiple and overlapping coding.

Figure 5



CoAn has an integrated editor for the definition and maintenance of dictionary entries. However, this does not support the creation of new categories.

3.5 Text and coding exploration

CoAn can generate an index of all words of a text. This can be filtered either according to word length, that is the user may specify how many characters a word should consist of and only those words whose length is at least the specified minimum number of characters and up to the specified maximum number of characters will be included in the index, or according to the total of frequency of occurrence of each word, sorted according to increasing or decreasing frequency. Alternatively, the word list may be sorted by endings. With CoAn word comparison lists can be produced for two different texts.

CoAn does not support overall text searches for a word or a string of words. The user can obtain a KWIC display, but only for a word or a phrase of the dictionary used. In fact, KWIC displays are possible only after a dictionary has been specified and the content analysis has been performed. In general, as CoAn supports search based on categories

rather than general text, it does not provide any assistance for text exploration and dictionary development.

CoAn generates text statistics information about both words and characters contained in each word, such as number of types, number of tokens, etc., and provides basic graphic presentations of type token statistics and coding.

3.6 Export formats and interoperability

CoAn exports the results of coding as an SPSS syntax and raw data file which contains the frequency of occurrence for each category. They are calculated for the higher level of the text structure specified. Furthermore, CoAn may write to a file the sequence in which categories occur, e.g. Category1 followed by Category7 for the ID1. In addition, the coded text can be generated in HTML format with the coded segments displayed as links.

3.7 Discussion

CoAn is a basic computer-assisted content analysis program. It supports both fully automatic as well as interactive coding of a text file which is based on an user-provided dictionary. The coding results can be exported in the appropriate format for further statistical analysis with SPSS. An interesting export option in CoAn is that coding results may be exported/saved as HTML files.

Apart from generating some basic text statistics, word indexes with frequency of occurrence information, and category-based KWIC lists, CoAn hardly supports text exploration and the development and enrichment of the dictionary.

Although it is a Windows program, modern GUI technology is minimal. CoAn gives rather the impression that it is a collection of single input-in output-out routines organised under a Windows interface.

3.8 References and further information

More information about CoAn (including the downloadable demo) can be found on: http://gwdu19.gwdg.de/~mromppe/soft/soft.htm. CoAn comes with a tutorial which guides the user through the main operations of the program.

4. Code-A-Text MultiMedia system

Code-A-Text is a Windows program[4] which was originally designed to analyse psychotherapeutic conversations. Nevertheless, it may also be used for analysing other in-depth dialogue texts as well as process notes and for coding the answers to open-ended survey questions.

Texts in Code-A-Text can be in audio, video or transcript form and the program allows the user to combine audio recordings and transcripts. Some combination of qualitative and quantitative analysis methodology is supported. The program provides basic word counts for whole texts or selected sections. Codes can be converted to a numeric format and written into a command file that can be read by SPSS.

Coding in Code-A-Text can be performed both automatically and interactively. Coding can be based not only on a dictionary of words and phrases but, also, on user-defined nominal or ordinal scales.

For our review we have used the trial version of the Code-A-Text MultiMedia (V3) system.

4.1 Terminology

Active Words: Words defined by the user as relevant and meaningful for the purposes of analysis.
Archive: It contains a dictionary of all the words known to the project, a list of all the Active Words and a list of the Word Groups.
Categorical Scales codes: User-defined codes which contain a set of non-numeric values.
Coding Frame: A group of database tables which store the project information like the Coding Manual and the texts to be analysed or which have been coded.
Coding Manual: A 'database' storing speaker identifiers, names of valid scales, and code names for each scale.
Content code: An Active Word whose occurrences in text are coded.
Interpretation codes: User annotations, i.e. memos.
Numeric Scale codes: Codes with numeric values. These are computed by the program taking into account the frequency of occurrence of particular words of a text segment.

---

4 Code-A-Text has been developed by Alan Cartwright, a psychotherapist, previous director of the Centre for the Study of Psychotherapy, University of Kent at Canterbury, UK, and now Senior Lecturer in Psychotherapy in the Kent Institute of Medicine and Health Sciences at the University of Kent.

Speech Unit: A text structure unit corresponding to dialogue turns which are designated by a special delimiter. A Speech Unit contains at least one Text Segment. For non-dialogue texts a paragraph is a Speech Unit.
Text Segment: is either a dialogue turn, in which sense it is a Speech Unit, or a smaller part of a speech unit, as defined by the user. A Text Segment is the coding unit in Code-A-Text, i.e. the unit to which one or more codes are assigned.
Word Groups: User-defined content categories which contain Active Words. For example, the Word Group 'anxiety' may consist of the following words: anxiety, avoidance, churn, churning, doubt, worried, worry and worrier.

4.2 Text importing and structure

Text can either be typed directly in the Code-A-Text editor or opened as ASCII or a RTF files. Each text corpus is subdivided within Code-A-Text: in dialogue texts the main subdivision is the Speech Unit. This in turn is determined by the Speaker Designator which is a special character. A Speech Unit contains at least one Text Segment which is a speaker turn for dialogue texts and a paragraph for non-dialogue texts. Code-A-Text can further split each Text Segment into sentences automatically. The Text Segment is the structural element on which coding is based.

Code-A-Text 'parses' a text and creates a database of the texts and its coding and a word index. The program can parse text successfully only if the Speech Units and Text Segments are delimited according to program-specific 'rules'. The user must either pre-process the text before importing or edit the text accordingly in Code-A-Text before parsing it. The system contains a Check Text before Parsing function which will identify most errors and inform the user how they should be corrected. Wrong format will be indicated by illegal segment error messages during parsing.

Texts created in the Code-A-Text editor or imported can by edited in the Text Manager Window. The Format menu contains a number of options for processing the text using different fonts, type sizes, colours, etc. as indicators of particular features, for example for displaying a text segment which has already been coded. Note that Code-A-Text considers this possibility as a different art of coding, in the sense that the user may 'annotate' text segments by rendering them in different fonts or colours.

Code-A-Text supports the import of dictionaries. The user first tells the program about the dictionary format depending on whether the dictionary entries are, for example, words on a single line or a series of paragraphs containing words. If the dictionary comes in the form of groups of words these can be entered as Word Groups by means of using the Parsing option for each single group.

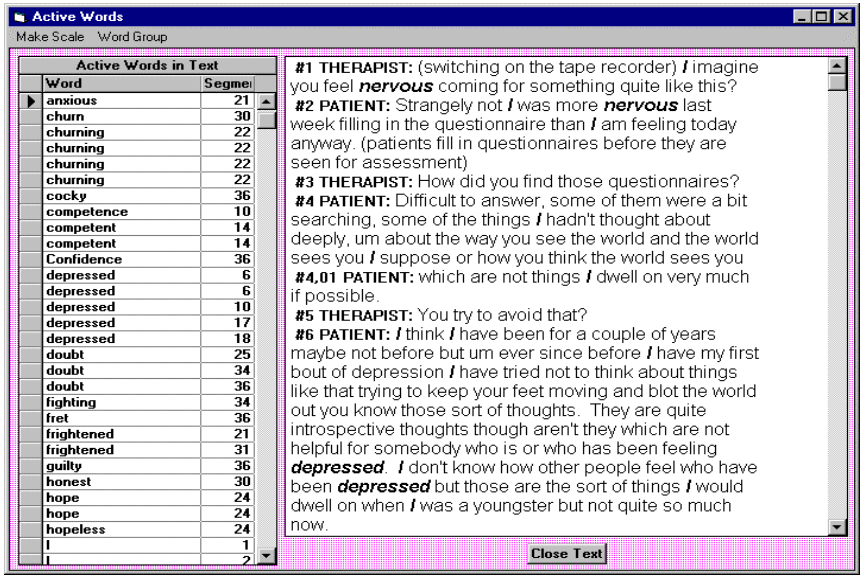
A Code-A-Text Transcription System, which is a standalone application, exists for the recording and transcription of sound files.

4.3 User interface

Code-A-Text combines basic word-processing and text analysis functionality in a single application. The Format menu contains the typical word-processing options for editing and formatting a text, whereas all the other menus are relevant only once the text has been imported and parsed.

The Code-A-Text desktop is organised according to a number of separate windows which all have their own menus or operation buttons and some of which invoke other windows. There are two possible text displays: text is displayed either in the main Code-A-Text window when it is first created, imported or during editing, or after one or more Active Words (see section 4.1) have been selected. The latter display (as shown figure 6) produces a new window, called Active Words window, which shows side by side a list of the selected Active Words and the context in which the words occur for a whole text. The Active Words in text are shown in different font type.

Figure 6

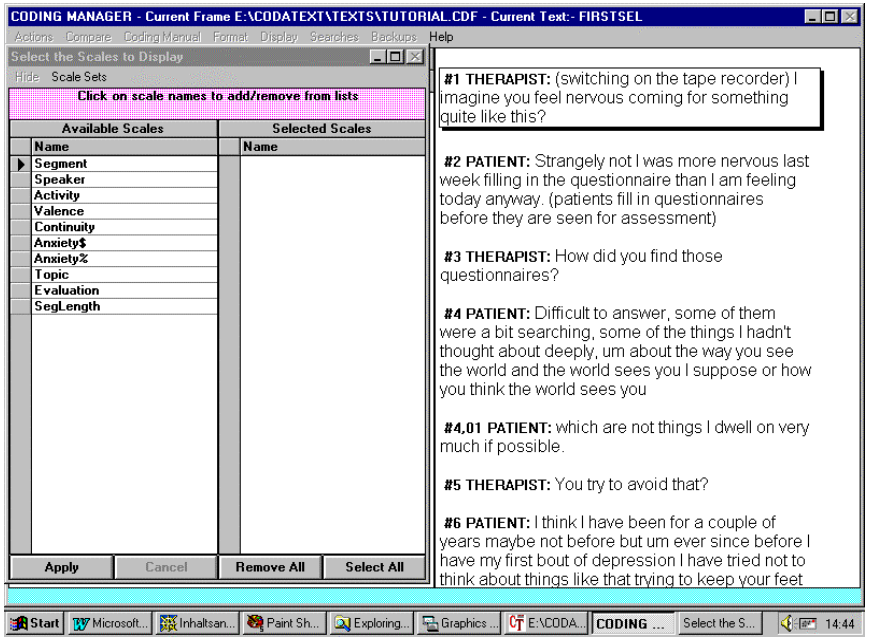


The Active Words window displays all the Active Words and Word Groups listed in tabular form. Operations such as creating, importing, and merging words and Word Groups are performed in this window.

All the words of an Archive are displayed in the Word Viewer window. Note that words may be moved between archives between Coding Frames; in that case the total of words may be higher than the total of words of the texts belonging to a Coding frame.

The Coding Manager window, as shown in figure 7, displays a list of all codes and their respective text segments. Operations such as the definition of scales and interpretation codes as well as of the different settings for formatting and marking up text can be performed in this window.

Figure 7



The list of codes includes all selected scales regardless of whether or not these have been used for coding. As this list can get fairly long, the consequence is that coding becomes confusing. However, the user can define which scales to display and save sets of scales so

they can be chosen and used without having to select the individual items. The user can also choose the order in which scales occur.

The Select Text window is used for filtering those texts over which a search should be performed. The definition for a search concerning coding in Code-A-Text requires a specification of how the result should be presented; a dialogue window, whose design is rather complex, is invoked when the Search Output option of the Output menu is selected.

The Scales Analysis window contains options for estimating the strength of association between any code and all other codes including content codes in the coding frame.

The Media Editor window is an audio editor window: the user can transcribe text from sound files, edit existing texts, and code segments of text as they are entered into Code-A-Text. Many of the features of the media editor are superseded by the Code-A-Text Transcription System.[5]

The system is strongly menu-driven using neither shortcuts nor icon buttons. In order to perform an operation the user has either to (i) choose the relevant menu option or (ii) choose the relevant menu option and then set a number of further options or parameters for the required operation (iii) or perform the previous two steps and then establish a new set of options or parameters based on the previous choices. Although the complex interface is partly due to the fact that Code-A-Text *does* offer varied and rich functionality, a clear and compact interface design would make the system easier to use and really enhance the system's functionality.

A useful possibility when working with the text in Code-A-Text is that the user may set bookmarks in text which facilitate moving around the text.

The on-line help in Code-A-Text contains step by step instructions for each topic which are then illustrated through a video. The help texts, however, are not always clear enough or do not describe the exact process for an operation.

4.4 Text coding and categorisation

Code-A-Text enables the user to define different 'types' of categories in order to code text segments. In addition to user-defined categories, Code-A-Text supports the automatic creation of Numeric Scales based on frequency of occurrence information.

---

5 Note that we have not tested this functionality.

Coding may be performed either by means of

- user-defined Categorical Scale codes, each of which contains a set of non-numeric values or
- similar to content analysis by coding each text segment on the basis of Active Words and Word Groups or
- Numerical Scales which are based on the frequency of occurrence of particular words of a text segment.

Before coding the text by Scale codes these and their respective values must be defined in the Coding Manual. Each scale consists of a user-defined number of codes and may entail comments about the meaning of the different codes. After the codes have been defined, text coding may start from the Coding Manager. Coding can be restricted to a selected number of the scales or can include all scales. Each text segment can be coded with one value of a scale. If a Speech Unit contains too much information to be coded in one step, the unit can be divided into sub-units which may be coded separately. Note, however, that Code-A-Text does not enable overlapping coding.

The user may select from a complete word list all 'meaningful' words and define them as active. Code-A-Text allows you to combine words to Word Groups, similar to dictionary entries for automated content analysis. The user may create Word Groups out of words with specific forms, e.g. words ending in '-ing' or '-ly'. For this purpose, Code-A-Text supports searches for strings of characters forming either the beginning or the end of a word or occurring anywhere within a word. The user can then select from the retrieved list of words those which should form a Word Group. Both Active Words and Word Groups can be converted into new scales.

Another type of codes supported by Code-A-Text are Numerical Scales codes. These codes are calculated based on the frequency of occurrence of specific words of a text segment and are automatically assigned to the text segments.

Code-A-Text supports the import of Scale codes into a Coding Manual. These Scale codes, however, must belong to another Coding Manual which has been created using Code-A-Text.

Finally, each text segment may be commented by an Interpretation which in principle parallels memo writing in other programs.

4.5 Text and coding exploration

The user can search for a word or a string of words in the whole text or a subset of it. The text positions in which the word or the string of words occurs are then shown in colour. Searching for a word in memos is also possible in Code-A-Text. The results of these searches can be saved for later inspection. These search definitions may not be saved and re-used.

Four different types of search are available:

- text (word) search,
- Categorical Scale search,
- Numerical Scale search, and
- text search in interpretations.

Boolean operators relating different words or Word Groups or Scale codes can be used for defining a search. In Numerical Scale search a relationship between the code (i.e. value) of scale and a numeric value is checked. This means the value of the code is checked, for example, whether it is less, equal or greater than a specified value. In defining a search the user may combine a Categorical Scale search with a text search.

Another type of search in Code-A-Text is predicative search which searches whether in a specified context a code occurs following another code. The user provides two Scale codes, the system searches for the first and puts an 'anchor' on the first hit, asks for the amount of text segments in which the search for the second code should be made, and then looks repeatedly for all positions in which the second code follows the first one within the amount of text segments specified.

The result of a search can be stored as a table, as a report or as filtered text, or as a new scale, which means that a new scale will be generated with the codes (values) True or False. For a hit in a text segment the Scale Code for this segment is True, otherwise it is False. The new scale can be stored and used for further searching.

As far as searching is concerned an important feature of Code-A-Text, is that the user may define complex searches where a combination of codes, Active Words or Word Groups is possible. The program keeps a log of these definitions so that they can be used again. The range for a search can be a speech unit or a text segment. In addition, the user can specify whether the search should be over all speakers - if dialogue texts are analyzed - or only for a particular speaker. The user may specify the amount of Text Segments to be considered for a search.

Code-A-Text offers different kinds of frequency information: general frequency of occurrence counts for all words or a selected word or Word Group, frequencies of Word Groups per text segment, the association of the scales to selected words, and a sequence of word analysis.

Furthermore, Code-A-Text, calculates some statistics for Categorical Scales, such as scale frequencies, cross tabulations, scale comparison, and numeric statistics e.g. mean, standard deviation, minimum and maximum values, etc. The Scale Analysis option allows you to trace associations within and between specified Scale codes or Active Words and Word Groups within texts. An option for a Hierarchical Scale Analysis provides a tree of associations with a single set of scale selections. It lists the Scale codes which are most highly correlated with the dependent Scale codes and then it displays the codes which are most highly correlated with these scales.

A kind of Key-Word-In-Context display is provided with the display in the Active Words window: a list of selected Active Words are displayed highlighted in context for the whole text.

Code-A-Text also supports a kind of sequential word search. The user specifies a word, e.g. the word *I*, and the programme retrieves every sentence containing this word and then groups these together. For example, it will group together all sentences containing *I am*, then all containing *I should*, etc.

4.6 Export formats and interoperability

The output of a search can be stored as an ASCII or RTF file, as a table or as an SPSS syntax and data file. If stored as an SPSS-readable file, all scale codes are converted to numeric values for further analysis with SPSS.

However, the syntax and data file produced by the version we used caused problems. If a scale contains no value for a text segment, there is a blank for the code in the file produced, but the data are stored in 'free' format and the blank functions as value separator in the 'free format'. This means that all the data following the blank get out of place when reading the file with SPSS.

4.7 Discussion

Code-A-Text is one of the few programs which cannot be categorised as a program for strictly qualitative or quantitative text analysis. It offers a fairly large variety of features in comparison with some of the other programs reviewed.

The design of the user interface of Code-A-Text may appear confusing and requires from the user to invest considerable effort to familiarise herself with it and use the software

efficiently. This is especially so for users with fairly limited experience in using text analysis software. A more efficient and compact organisation of the numerous menu options in the numerous windows would improve working with Code-A-Text. We found the search procedure and the design of the search output window difficult to learn and use mainly due to the fact that a very large number of options is 'clumped' into the same window. A characteristic of the user interface design deficits of Code-A-Text is that there is no systematic positioning of such standard operations such as closing a window. Moreover, the user needs to master the Code-A-Text 'terminology' to understand what is possible in the system and how to apply it.

One restriction of Code-A-Text is that it does not support overlapping coding. However, a way round this restriction is to create a 'coding hierarchy' which will allow coding of two segments with two different codes.

The search functionality offers a large variety of options as well as combinations of options, but the actual steps involved in defining and starting a search are not straightforward. In addition to that, the search definition procedure can become tedious since the user must first define a search, then - in a new menu - define the type of output to be produced, then select a text to be searched, and finally start searching by selecting the search type.

Finally, Code-A-Text comes with a number of features concerning the synchronisation of text and sound. For example, the possibility to add pictures to text segments. Thus enables the user to attach a frame from a video or a graph from the table output window to a text segment or the coding of video or sound data without prior transcription. As mentioned in the first part of the book, however, this review focuses on text material only. If a program provides multimedia functionality this is reported but has not been reviewed.

4.8 References and further information

The user version of Code-A-Text MultiMedia System is supplied on CD ROM and comes with the MultiMedia Workshop which provides a comprehensive introduction to the programme. Also, the user version comes with a help system which has a number of videos illustrating the tutorials.

More information to Code-A-Text is available from the following Web site: http://www.codeatext.u-net.com/. The program is distributed by Scolari, Sage Publications http://www.scolari.com/code-a-text/code_a_text.htm.

5. DICTION 4.0

DICTION is a Windows program which was developed by Roderick Hart at the University of Texas and is distributed by Scolari Software of Sage Publications. It has been created for the analysis of short verbal messages and especially of those of political discourse, such as presidential speeches.

DICTION attempts to determine the language characteristics of texts on the basis of calculated scores for five general categories, namely Activity, Certainty, Commonality, Optimism, and Realism, comprising twenty-five sub-categories. The program incorporates dictionaries for each sub-category with a word list characteristic of each sub-category. The result of analysis with DICTION is a report about the specific text it has processed and a numeric file for statistical analysis. We review the demo version of DICTION 4.0.

5.1 Terminology

Dictionaries: Twenty-five word lists for each DICTION 'sub-category', e.g. 'Tenacity', 'Collectives', 'Numerical Terms'.
Report Extrapolations: An option for working with texts whose size is less than 500 words and which makes corrective counts for standardising them to a 500-word basis.

5.2 Text importing and structure

DICTION reads ASCII text files whose structure follows the format expected by the program:

- the text files must contain carriage returns,
- in the event of some descriptive text which is not to be considered for analysis, called *descriptive identifiers*, which should not be considered in the analysis proper, e.g. titles, author, etc., this text is contained within special characters,
- any *alpha-numeric* variables defined by the user which should be considered for the statistical processing of the text data are also delimited with a special character.

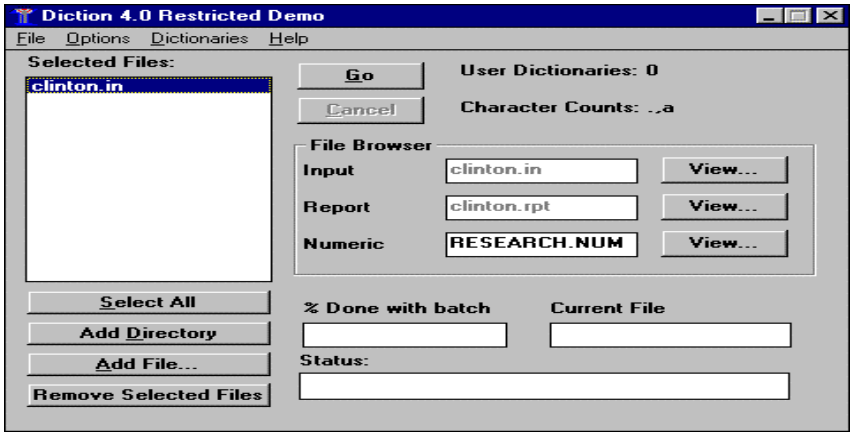
DICTION is case-insensitive and therefore accepts files in either upper- or lower-case. Texts, dictionaries, and output files must comply with the prescribed directory structure and file extensions.

The scoring mechanism of DICTION is sensitive to file size and the program can process input files of maximum 500 words. It analyses either only the first 500 words of a given passage or any text up to 5,000 words. In the latter case, DICTION will automatically break up the passage into 500-word segments. Texts shorter than 500 words can also be processed as Report Extrapolations which makes corrective counts of a small file, thereby standardising it to a 500-word basis.

5.3 User interface

As shown in figure 8, the DICTION desktop is simple, consisting mainly of one window comprising operation buttons for file selection, dictionary-based analysis, and browsing of the texts, the reports (text and analysis results), and the numeric file.

Figure 8



The menu bar accommodates the options for browsing the dictionaries and setting a number of parameters for analysis, e.g. whether texts with less than 500 words are used or the kinds of information to be included in the report.

DICTION displays on a separate window the words composing each sub-category, displaying only one category at a time. A drop-down menu lists all sub-categories.

The user can specify which editor should be used for browsing the texts, the reports, and the numeric files.

DICTION does not support the communication between two different windows and the user cannot select an item on a display (e.g. dictionary entry or text segment) and perform an operation on it. On-line help is organised either according to content groups or as an index of keywords.

5.4 Text coding and categorisation

DICTION analyses the words of a single text of maximum 500 words based on the program's own dictionaries which are organised as word lists. There are twenty five such

words lists, e.g., the Communication one contains words such as 'advice, recommend, urge'. The program matches the words in the lists with the words of the analysed text.

The DICTION dictionaries are specific for the types of frequent vocabulary used in American political discourse. The twenty-five word lists fall under five more general categories, namely Activity, Certainty, Realism, Optimism, and Commonality, which are claimed to be 'the main' dimensions or features for describing the language of political speeches (Hart 1985: 111). The word lists are, hence, 'judgmental' since they involve, on the one hand, the developer's own estimations of the way political discourse functions and, on the other hand, his beliefs about which features are crucial to the type of language analysed. There is no empirical evidence provided for the validity of this categorisation and its respective word lists for the analysis of political language.

The user cannot edit, enrich or modify in any way the DICTION word lists. Up to ten user-custom dictionaries may be added to the existing DICTION word lists and each of them may contain a maximum of 200 words.

The result of the matching of the words in the lists with the words of a particular text is the calculation of a number of scores for raw frequency counts and comparative data based on texts which have been previously examined by DICTION. In particular, the comparative data produced are based on a sample of 20,000 public texts including political speeches, press conferences, patriotic ceremonies, etc., as well as broadcast news, journal reports and editorials, citizen complaints, corporate reports, religious preachment, etc. Note that the sample texts were produced in the United States between 1945 and 1996. The purpose of the sample is to assist the user to 'locate' the particular text analysed in the larger universe of discourse.

On the basis of scores for the individual sub-categories, DICTION calculates a standardised score (normative data) for the five general categories. This calculation involves translating raw dictionary totals into z-scores, adding and subtracting them "in the appropriate fashion" and then adding constants of 50 to eliminate negative numbers. For example the formula for calculating the score for the one broad category Certainty is:

Certainty = [Tenacity + Leveling + Collectives + Insistence Score[6]] - [Numerical terms + Ambivalence. + Self-reference + Variety Score[7]].

---

6 This is a measure for calculating a text's dependence on a limited number of often-repeated words. In calculating this score, DICTION singles out all words used three or more times (in a 500-word text).

7 The Variety score provides the type/token ratio.

It must be noted that the five categories are not statistically related: knowing the score for the one does not help estimating the score for the other.

Some of the words included in the DICTION word lists are assigned to more than one of the five categories; for instance, 'community' contributes to both Certainty and Realism scores. DICTION deals with homographs by using word frequency information. The counts for homographs were differentially weighted within the various dictionaries by applying (a priori) statistical norms for use of those terms (Hart 1985: 110).

As it may have become obvious, DICTION does not code text segments of a text. Rather it categorises a text according to a number of variables which are defined specifically for the analysis of political language by the developer of the program.

5.5 Text and coding exploration

Beside the numeric file with the calculated scores, DICTION provides a report including some text statistics, such as the number of words analysed, number of characters, number of word types or average number of characters per word. A list with the frequency for each character as well as a word list of all words with highest frequency of occurrence equal to or higher than three times are also included in the DICTION report.

The report contains also the raw scores, the comparative data, and the standardised scores (as explained below). Furthermore, four additional scores are provided:

- Insistence score: a text's dependence on a limited number of often-repeated words
- Complexity score: the average number of characters per word
- Embellishment score: the ratio of adjectives to verbs, which is based on the assumption that "heavy modification 'slows down' a verbal passage by de-emphasising human and material action", and
- Variety score: the type/token ratio.

The calculation of each of the above-four scores takes into account a small number of sub-categories and is relevant to one of the five general categories. For example, the associated word lists for the Embellishment score are 'Praise', 'Blame', 'Present Concern' and 'Past Concern' for the general category 'Activity'.

The totals for user-customised dictionaries are also reported if those are used.

Not supported are searches over a group of texts for a word or a string of words or a 'category', i.e. all occurrences of the words included in one of DICTION's the word lists.

5.6 Export formats and interoperability

The output of DICTION consists of two files: the report (as described above) and the numeric file. The latter contains variables such as words/text, characters/text, word types, or the scores for each sub-category and each of the five main categories. DICTION provides an SPSS syntax file which can be used to import the data into SPSS.

5.7 Discussion

DICTION is specifically designed for elucidating the rhetorical characteristics and style of political discourse. In order to use DICTION the user must accept the theoretical, categorisation, and scoring assumptions it makes. The dictionaries DICTION bases its analysis on cannot be modified by the user. Although the user can use her own additional dictionaries, she has no control over the categorisation and, hence, the interpretation mechanism which is integrated in DICTION. Evidence for the validity of the word lists used for the analysis of political language is not provided.

Although DICTION is advertised as an all-purpose program whose dictionaries have been designed for use with any sort of English-language text, the norms that come with the program are based largely on political and mass media text materials. As stated by Hart (1985: 110) DICTION "is the only extant program expressly built to accommodate political messages; its various dictionaries are concerned with the types of words most frequently encountered in contemporary American public discourse."

In DICTION the size of each text analysed is limited to 500 word per 'text passage', that is, per analysis run. It is not at all clear why this text size is chosen. What difference would it make if 702 words were allowed instead? The definition of high frequency for a word is that it occurs more than twice. Again, it is not clear why this decision was made.

DICTION focuses on matching the words of a text according to its dictionaries and producing appropriate scores for interpreting the text's features. Text exploration operations and categorisation scheme maintenance are not really supported.

5.8 References and further information

Further information about DICTION (including the downloadable demo) can be found at: http://www.scolari.com/diction/Diction.htm.

DICTION comes with a User's Guide. A description of DICTION together with an example of applying it is given in Hart (1985).

6. DIMAP, MCCA and MCCALite[8]

DIMAP[9] is a dictionary development software which comes with a content analysis component, the MCCA, as an integrated part of the software. A 'lite', standalone version of MCCA, called MCCALite, is also available. This has a newer graphical user interface, but provides less of the MCCA functionality of the DIMAP beta version, and, moreover, it does not feature the advantage of combining a dictionary development and maintenance program with content analysis, something that is realised by the combination of DIMAP and MCCA.

Input in MCCA can be transcripts of speech or interviews, answers to open-ended questions, or other free textual material in English language only.

MCCA supports quantitative (automatic) content analysis based on a dictionary comprising 116 categories. The differentiating aspect of content analysis with MCCA is that the automatic categorisation (coding) per se is not the required analysis outcome, but rather a by-product of the analysis. The user of MCCA aims at obtaining Emphasis-scores and Context-scores for a given text corpus, which can be further statistically analysed and which are indicative of the content of the texts.

Our basis for reviewing the Dictionary Maintenance Programs (DIMAP) has been the DIMAP-3 beta version for Windows. This includes the MCCA content analysis program as well as the standalone MCCALite.

6.1 Terminology

C-score: one of the two types of the scores provided by MCCA which consists of a profile of four social contexts scores: traditional, practical, emotional, and analytic.
E-score: stands for emphasis score. One of the two kinds of scores that are the result of the MCCA processing, which represents the emphasis placed on each of many idea categories.
Idea category: consists of a group of words that reflect a given idea or meaning. For example, the idea of 'control' occurs when words such as 'allow', 'authorize', 'insist', and 'command' are used. The MCCA dictionary distinguishes 116 idea categories.

---

8 MCCA stands for Minnesota Context Content Analysis and was developed by Don McTavish. MCCALite was ported into Windows by CL research. Throughout this section whenever we talk about text analysis we refer to the MCCA component, whenever we refer to dictionary construction and maintenance we refer to DIMAP-3.

9 DIMAP stands for Dictionary Maintenance Programs and is a commercial product sold by CL Research.

6.2 Text importing and structure

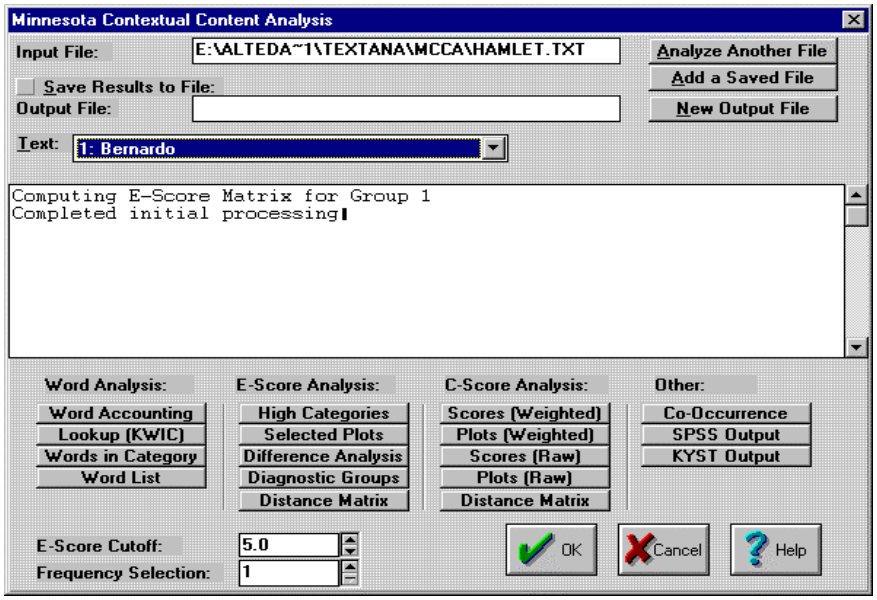
MCCA analyses ASCII text which is not 'imported', but rather read against the MCCA dictionary. Text must be pre-processed to delimit with a special character (i.e. '=') text titles or speaker names. By enclosing it in square brackets, the user may delimit text, which should not be considered for content analysis.

Texts cannot be edited while using the MCCA content analysis option.

6.3 User interface

Content analysis is a separate option contained in the DIMAP Resources menu and integrates the MCCA program. The user must specify the name of the file to be analysed and provide the text structure delimiters used before starting to process the text file according to MCCA. When the text file has been read against the dictionary, four different groups of information are presented, namely Word Analysis, E-score Analysis, C-score Analysis and Other (see figure 9).

Figure 9



Each of these contains a variety of click-on buttons for obtaining the results of particular types of analysis, such as word frequencies, C-scores, C-scores as plots, etc. These results are displayed in a small and non-extensible text-box, making it difficult for the user to see the results as a whole. Moreover, the plots are not displayed as graphics but rather as 'texts'. The MCCALite user interface is an improved version of the DIMAP content analysis window with a more user-friendly display of the analysis results. In general, both DIMAP and MCCALite interfaces are simple and functional.

To analyse the content of a text according to the MCCA dictionary, the user specifies a text file and sets the appropriate options for the text delimiters used. MCCA does not support text browsing: the user has no access to the original text from the MCCA window nor to the 'coded' text. However, a solution to this restriction is to use the Tag option in DIMAP to code a text according to a particular dictionary and save the result as an ASCII file. This is the only way the user can view the coded text. The file contains one word per line followed by one or more code numbers. The headers of a text are also 'tagged' and those words which have not been found in the dictionary are assigned the category 'unk' for 'unknown'.

The MCCA dictionary may be maintained in DIMAP: once the dictionary has been opened from the DIMAP desktop, a list of all its entries is displayed. Double-clicking on an item of the list displays its definition properties. The user may use a number of click-on buttons for creating new entries, editing existing ones or deleting an entry. The creation and editing buttons bring up a new window with a number of fields for specifying the properties of an dictionary entry: its name and number, its senses, its place in a hierarchical organisation, as well as its semantic features and syntactic behaviour.

DIMAP provides general on-line help as well as window-dependent help which not only describes the way the program can be used, but also provides the details of some of the processing involved. Some of the help texts, e.g. disambiguation process, are difficult to understand. The texts would considerably profit from providing concrete examples, e.g. for a semantic restriction rule or a template.

6.4 Text coding and categorisation

MCCA codes English texts automatically on the basis of a dictionary. The MCCA dictionary includes slightly over 11,000 words. According to McTavish (1997: 381) this accounts for 90% of English usage, categorised according to a total of 116 categories.

The MCCA (and MCCALite) analysis process first reads a text file and then locates those words in text which are included in the MCCA dictionary and categorises them accord-

ingly.[10] The frequency of occurrence profile of each word is compared against that for general American English use. This process forms the basis for the MCCA output, which consists of two separate score profiles: the E-scores, which capture the ideas that are emphasised (note that the ideas are the dictionary categories) and the C-scores, which capture the structuring of ideas, which may determine different social contexts.

MCCA was developed for American English with standard usage probabilities for this language and therefore is language-dependent. The words contained in the MCCA dictionary also contain information about their expected frequency of occurrence. The expected frequency of occurrence probabilities which provide information about the 'standard' usage of a word are based on the work of Kucera/Francis (1967) for American English. These are available for calculating the E-scores.

E-scores show the emphasis placed on each of the MCCA idea categories. Given that E-scores are 'normed' against the expected usage of the words in an idea category, positive E-scores indicate a prominence of an idea category whereas negative E-scores may signal absence or insignificance of an idea category in the text. Normed scores are computed in a z-score-like fashion, contrasting category proportions with the expected probability of use of a given category. These are divided by a standard deviation of expected category usage across four social contexts (see below).

Four institutional themes are used for calculating the C-scores. As described in McTavish et al. (1997: 173) these are:

- *"Traditional* - a focus on norms and expectations for appropriate behaviour; this is especially emphasised in text from judicial and religious organisations.
- *Practical* - a focus on successful (efficient) goal accomplishment; this theme is most evident in business and work organisations.
- *Emotional* - a focus on personal involvement, comfort, enjoyment, or leisure; this theme is typical of leisure or recreational activities.
- *Analytic* - a focus on objectivity, curiosity, or interest; this emphasis is more pronounced in research and educational settings."

The profile of four C-scores provided by MCCA for a text measures how closely the pattern of the language of a particular text matches each of these.

---

[10] Note that numbers and non-alphabetic, non-numeric characters are *not* yet included in the lexical analysis. This is similarly true of proper nouns, which usually end up in the list of unknown words.

The categories of the MCCA dictionary were developed "judgementally" (McTavish et al. 1997: 172). Some of the categories are syntax-oriented and others reflect a given idea or meaning. Words may be assigned to more than one category, in which case the dictionary entry has more than one sense. The words of a text are, however, assigned *uniquely* to one category (this also includes the category for the left-over, i.e. non-categorised, words). Note that the non-categorised possibility is not considered during the disambiguation phase.

One of the most interesting features of MCCA is that ambiguous words are contextually disambiguated to determine the most appropriate category and that each category receives some differential weight reflective of its usage in the four social contexts. Each word that is categorised in a text contributes to a running context score that is used for disambiguating words that have more than one sense. MCCA uses weights which reflect the relative use of each (idea) category in the four social contexts. In particular, each of the categories has an estimated frequency of occurrence in the four contexts. These frequencies were estimated by factor analysis and "experience". After averaging these frequencies across contexts, the percentage for the deviation of its frequency from the average is computed.

To compute the C-scores one must have disambiguated beforehand and the disambiguation takes place while the C-scores are calculated. The running context score is used absolutely before a context emphasis score is well established and this may result in disambiguating incorrectly.

As mentioned above, if MCCA is used within DIMAP, the MCCA dictionary may be extended or modified as supported by DIMAP. The user cannot develop or maintain the dictionary if the MCCALite standalone version is used. Furthermore, support for dictionary validation is provided by DIMAP by means of defining searches for codes or superconcepts. Nevertheless, it is not possible to search the dictionary for all words having more than one sense, or all entries having semantic restrictions, or specific semantic restrictions. This would be useful for dictionary development.

6.5 Text and coding exploration

The Prescreen Input File option of the MCCA window generates a list with some general text statistics, e.g. number of word types, tokens, sentences, average word length, etc. After the text file has been read against the dictionary, four different groups of information are presented, namely Word, E-score, C-score analysis, and Other (with Export functionality).

The Word Analysis group comprises buttons for obtaining

1. some word statistics, such as the total number of words, percentage of word types, the total of words which has been categorised or word length statistics.

2. KWIC lists (note that the left and right context is hard-coded and cannot be expanded).
3. A word list with frequency of occurrence information, and
4. a list of words according to the category to which they have been assigned.

The E-score Analysis group comprises click-on buttons for obtaining different analysis results concerning the E-scores. Some examples are a category list with those categories having highest scores, plots for the 'thematic' (as opposed to syntax-oriented) categories, comparisons of E-scores for different texts, etc.

The C-score Analysis group offers a list of click-on buttons for obtaining different analysis results concerning the C-scores, such as raw or weighted C-scores. The user may obtain an easy plot (text type output) of the weighted scores or generate a distance matrix of the C-scores.

MCCA allows the use of a stop word list. The user can specify a frequent words list in order to speed up processing and possibly to eliminate the words on that list from inclusion in either or both of the context and emphasis scores. The user may run an analysis and modify the frequent words list based on a frequency analysis and re-run the program, choosing to exclude such words.

6.6 Export formats and interoperability

The export analysis group of the MCCA window consists of two different options, namely SPSS and KYST output, for exporting the E- and C-scores. For the SPSS output, an SPSS syntax file that includes variable names, specifications, and variable labels is provided. The output data consist of the following data elements: sequence number of the text, total number of tokens, total number of categorised tokens, number of unique tokens, number of unique categorised tokens, C-scores, the mean of all 116 E-scores, lowest and highest E-score, and range of E-scores followed by the 116 E-scores.

As far as the KYST output option is concerned, a file is generated that can be directly input to the KYST program. This comes with MCCA and is a multidimensional scaling routine for analysing the C-score and E-score distance matrices. Unfortunately, KYST runs only under DOS. The results of the multidimensional analysis with KYST are stored in an ASCII file (which includes Fortran control characters for form feed).

Furthermore, the distance matrices of E-scores or C-scores can be saved to a file directly from MCCA. The first two lines contain a header followed by the matrix including row and column numbers. This format needs to be edited if one wants to use these matrices as input in statistical packages, for example for multiple scaling analysis.

6.7 Discussion

DIMAP with its MCCA component is unique in supporting the connection of a content analysis program (MCCA) with a dictionary maintenance program; a feature which may profit dictionary-based content analysis. Still, as it stands at the moment, the user of DIMAP cannot make full use of this potential. One's own dictionaries may be imported into DIMAP as long as they are in a DIMAP-format. However, at the moment no solution is offered as to how to use one's own dictionary with the MCCA option. MCCA outputs E- and C-scores, and if the words in another dictionary do not contain such information about the expected frequencies of occurrence, no content analysis may be performed.

The work reported by McTavish et al. (1997) and Litkowski (1997) shows interesting directions and potential for using already available general lexical information, i.e. the WordNet lexical database (see Miller et al. 1993) for extending a dictionary, developing it further, etc.

User interface problems aside, (e.g. poor presentations of the analysis results, plots, etc.), there are two main points we would like to discuss regarding the MCCA approach. The first concerns the probabilities for the expected frequency of occurrence and the second the disambiguation process. Expected frequency of occurrence probabilities provide information about the 'standard' usage of a word and are based on the work of Kucera/ Francis (1967) for American English. These probabilities are used for calculating the E-scores. The question arising is how stable these probabilities are over time. In McTavish/ Pirro (1990: 262), it is claimed (in a footnote) that "[A] summary of a variety of US-based text tends to confirm the stability of these general probability distributions." The degree of comparability of standard American English usage of the 60s with that of today, however, would need to be checked.

With regards to the disambiguation process, firstly it is not possible to determine the plausibility of the process via straightforward way. At the end of the day, what is important is whether the provided scores correspond to one's hypothesis given a particular text. Remember that the aim of the MCCA analysis is not to code per se, but rather to extract the E- and C-scores which can be used for interpreting social behaviour. In that sense, one need not worry about the particulars of the disambiguation process but rather whether or not the scores work.

Secondly, the running context score is calculated on the whole text specified for analysis (the global context) and not on the local (sentence) context. We would expect that here the size of the text plays an important role. If one analyses answers to open-ended questions, for example, then the amount of each answer text is quite small. Furthermore, if the text

analysed consists of a corpus of answers, then the vocabulary used in some of the answers may denote categories which are not the most frequent in the whole corpus. In that case, given that the running context score is the global context, the disambiguation will probably not work correctly, as it is the local context that should be considered instead.

6.8 References and further information

McTavish/Pirro (1990) and McTavish et al. (1997) report on content analysis using MCCA and provide a short description of the program. Litkowski (1997) and McTavish et al. (1997) report on using available lexical information for facilitating category development and in particular on using WordNet for extending the MCCA dictionary.

More information about DIMAP with a detailed literature list on DIMAP and MCCA can be obtained from: http://www.clres.com/.

7. HyperRESEARCH

HyperRESEARCH is a qualitative-oriented text analysis program running under both Windows (3.1 or later) and Macintoshes and is distributed by ResearchWare, Inc., USA, http://www.researchware.com/.

HyperRESEARCH may be used for the analysis of all types of texts, e.g. as interviews, self evaluations, memos, etc.

The program supports on-line coding of text according to one or more codes. A sophisticated feature of HyperRESEARCH is its theory-building and testing module which supports searches for a combination of codes using the Boolean AND, OR, and NOT operators and retrieving texts with coded text segments with the specified combination of codes. Theory testing is supported by 'IF ... THEN' rules created by the user. Reports with the results of searches on codes can be generated.

We review here the Windows demonstration version of HyperRESEARCH 1.65. The demo version is the same as the full user version, with the single limitation that the user cannot save her analysis.

7.1 Terminology

Case: The basic unit of analysis in a HyperRESEARCH Study. It contains the codes and the coded text segments for one or more text documents.
Source: An ASCII/ANSI text file. One or more Sources may be opened and coded for a single Case.
Study: The entire text analysis project. It contains the Cases, codes, and all references to Sources. A HyperRESEARCH Study consists of at least one Case.

7.2 Text importing and structure

To start a project in HyperRESEARCH, the user creates a Study which consists of one or more Cases. A Case is the unit of analysis in a Study and contains the codes and the coded text segments for one or more text documents. The text material to be analysed is stored in ASCII/ANSI text files, called Sources in HyperRESEARCH. The user may open one Source at a time when working for a specific Case to code it or display the coded text segments. Sources reside outside a HyperRESEARCH Study and they are, therefore, not imported in the program. The user may remove a Source already analysed within a Case. Editing a source file which has already been coded in HyperRESEARCH will therefore change the text which HyperRESEARCH associates with a given code and render it useless.

Because HyperRESEARCH cannot access more than 16,000 characters at a time (these correspond to roughly 3,000 words) the user must pre-edit long source files. These must be separated into 'pages' by placing a tilde on a separate line to mark a HyperRESEARCH page break before starting to code. HyperRESEARCH handles the pages of a long text as 'parts' of the same Source file.

Other internal text structure, such as paragraphs, lines, sentences, titles, notes or comments of speakers, etc., cannot be identified and is not supported by the program.

7.3 User interface

HyperRESEARCH organises text analysis tasks in three main windows called Index Card, Analyse Codes, and Hypothesis Testing windows. The first one offers an overview of the HyperRESEARCH Cases. The second supports searching for one or more codes in the Cases, obtaining information about the frequency of occurrence of one or more codes within one or more Cases. The third window supports the formulation and testing of hypotheses concerning existing coding. It is not possible to have more than one window open at a time.

The Index Card window comprises a tabular display of the following items: the Case Name, the Check column, the Source Type column, Reference column, and Code Name column. For each Case, the number of which is displayed in the Case Name field, all the coded text segments and their respective assigned codes are displayed. It is important to note here that the Index Card displays information for a *single* Case; the user may request to see the next or previous Case or go to a specified one. It is not possible to obtain a display of all the Cases of a Study with their respective coding and codes they have been assigned to. A way round this is to create a report that duplicates the Index Card window and Case information and either print that report or export it to a word processor.

The Reference column of the Index Card window displays information about the text segments coded, i.e. from which until which character of a Case it has been coded with the code assigned displayed in the Code Name column. If the same text segment has been coded with more than one code then this is shown on a separate line in the table. By clicking on the Check column the user brings up a dialog box with options for displaying the source text of the text segment coded according to the code checked, assigning an additional code to the displayed text segment, and renaming or deleting the code. The codes displayed in the Index Card window can be sorted alphabetically or by the their reference number, that is by the first character of a coded text segment included in the list. In sum, a number of different operations are possible form the Index Card window:

maintain the coding scheme, open a source text and code already coded text segments, as well as asking for a display of a coded text segment according to a specific code.

To code a text segment for the first time, the user opens a source text for a Case, selects a text segment on the window displaying the text and attaches it to a code. On-screen selection is possible by selecting (clicking on) the beginning and end of the word or string of words. The user cannot select and drag the mouse over a particular text position and release in order to select.

When a word or a text segment is assigned to a code the Index Card window is updated with the new information.

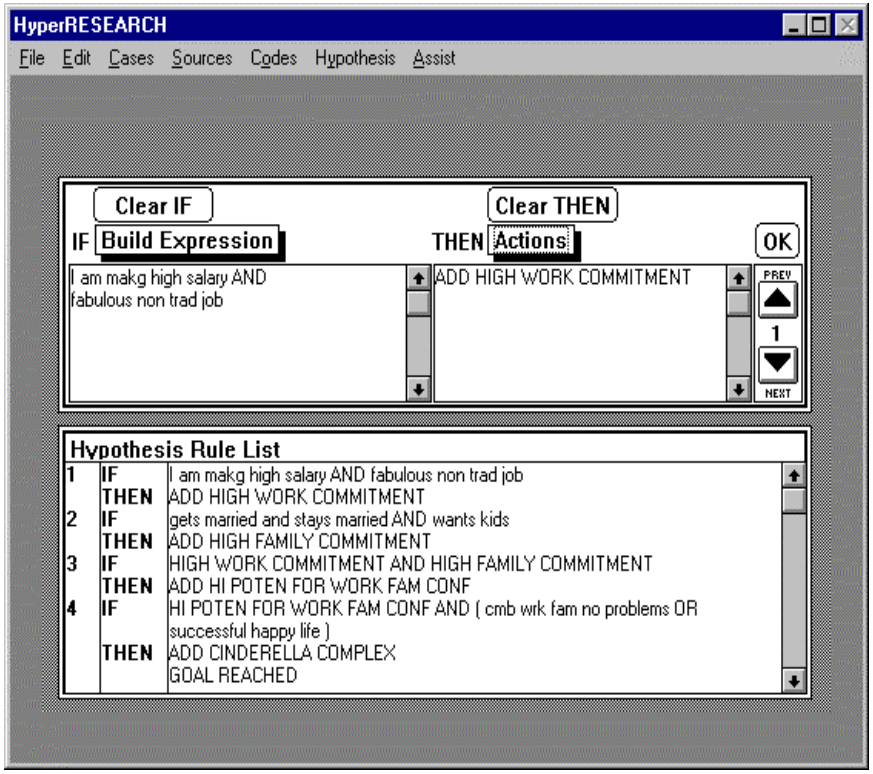
Unfortunately, HyperRESEARCH makes no use of graphical means (colour, different fonts, etc.) to indicate the coded text positions in a text. To see such positions the user must use the Index Card window, select the coded segment and hit the Display button. Note that only one coded segment may chosen at a time.

The functionality of the Analyse Codes window concerns the searching operations for the occurrences of codes in one or more Cases and generating reports which may be displayed on screen or saved as ASCII files. The window is organised in four quadrants each of which acts as a filter for specific options. The Select Cases quadrant specifies the Cases over which the search should be performed. When used in conjunction with the Build Expressions quadrant, HyperRESEARCH will automatically include or exclude Cases based on the presence and/or absence of certain codes. The Build Expressions quadrant makes use of the Boolean operators AND, OR, and NOT to show the relationships between two or more codes. The Select Codes quadrant allows the user to specify (by selection) which codes should be included in the report which HyperRESEARCH will generate. The Include in Report quadrant displays a number of check boxes for indicating what information should be included in the report and whether the report should be sorted by case or by code. To obtain the result of the analysis, the user chooses Display Report or Export Report.

The Hypothesis Testing window may be used for defining and testing hypotheses concerning a Study. A hypothesis is defined as a set of “If ..., then ...” rules applying specified actions to Cases when certain criteria concerning each Case’s coding are met.

As shown in figure 10, the Hypothesis Testing window is organised in three sections: the IF and THEN parts for a hypothesis definition and the Rule List section.

Figure 10



The IF section includes a Build Expression button for defining the underlying assumptions of each point in terms of the absence and/or presence of certain codes. In the THEN section the user specifies the inference portion of a HyperRESEARCH rule. The window includes an Action button for specifying what to infer if the conditions in the IF window are met. Completed rules are displayed in the Rule List. The user may keep building rules which are added to the list until a hypothesis is complete.

The on-line help is organised as an index of all HyperRESEARCH menu options. Clicking on an item of the help index displays the respective text for the selected option. The help texts are very short and, apart from a very brief definition, they provide very little ‘help’.

7.4 Text coding and categorisation

The user works with a single Case at a time when coding. She may select a text segment of arbitrary length and assign it to a code by clicking on the first and last characters of the segment. The selection brings up the Code Selection Dialog Box, where the user can type in the name of the code to be created. Punctuation marks may not be used for code names and the maximum size for a name is 32 characters. More than one codes may be created in one go if required.

The HyperRESEARCH Autocode option will code a specified word or a string of words automatically for a single Source, i.e. is the text file displayed at the time of using this option. Autocoding is not case-sensitive. This means that all occurrences of the string 'black' will be coded regardless of whether they are spelled as 'black' or 'Black'.

HyperRESEARCH supports overlapping coding. This is possible because the same text segment can be attached to one code or more. In addition, *all* text segments which have been assigned to a specific code may be coded with one code (or more). The HyperRESEARCH function Copy may be used for this purpose.

Assigning multiple codes to selected chunks of source material runs the risk of filling up the Index Card of a Case. There is a maximum limit of codes on a Case. This limit, however is based on many variables, including the size of the pages of the source text file. An Index Card may be full once the user has added anywhere between 200-500 code references per case when working under Windows. If this is the case, an error message notifies the user. The developers mention that the Windows and Mac versions differ on the number of code references they allow for a Case: the Macintosh version is usually able to handle 300-700 code references per case. If the code reference limit has been reached, the user can either consolidate her codes or begin a new Index Card by choosing New Case.

Codes may be deleted or renamed globally, i.e. over an entire Study. An additional global operation concerns the 'copying' of a code: all coded text segments of a selected code can be assigned to another code, which may already exist or be created during this operation.

HyperRESEARCH cannot handle code hierarchies. Also, the definition of any other relations holding between two codes is not supported.

Contrary to other qualitative analysis programs memo-writing is not possible in HyperRESEARCH. Furthermore, the program does not support teamwork or the merging of HyperRESEARCH projects.

7.5 Text and coding exploration

Existing coding may be explored in HyperRESEARCH either by making use of the Analyse Codes or the Hypothesis Testing functionality. The latter may be used to determine whether or not the coded texts, that is the available data, support a given hypothesis. The user can construct fairly complex queries on the coding by formulating 'If ..., then ...' conditional expressions. The results of such 'queries' are stored in a report for every Case stating whether the rules formulated hold or not. The rules built for hypothesis testing can be saved in a file and, therefore, be re-used – of course, only for analysis with HyperRESEARCH.

Searches are supported for the occurrence of codes in one or more or all cases by using Boolean operators and may be defined in the Analyse Codes window. The user can build expressions using the Boolean operators AND, OR, and NOT in order to check the relationships between two or more codes. HyperRESEARCH displays the results of the Analyse Codes operations as a report. Different kinds of reports may be generated depending on the options chosen, e.g. a report for the contents of each Case in the Study or a report for a Master Code list (i.e. a list of all codes currently in use in the Study with the frequencies of occurrence for each code, etc.).

In fact, the display of the result of searches either by using the Analyse Codes operations or formulating and testing hypotheses are always presented in the form of a report. This is displayed in a window which is not 'live', i.e. no further operations may be performed in it and the information it displays is not hyper-linked to other windows. HyperRESEARCH goes directly from performing an operation to producing a report file, which can be either saved or sent to the printer or displayed on screen. There is no 'intermediate' level of presentation and browsing, e.g. as a means of working with the results.

HyperRESEARCH does not support searches for either a word or a string of words or a word stem in a Case. Although the user can generate a report containing frequency of occurrence information with respect to the existing coding, i.e. how often a code has been assigned to a text segment, for how many Cases and for which Cases, information about the general frequency of all the words of a Case cannot be obtained.

7.6 Export formats and interoperability

HyperRESEARCH will save reports on the output of the searches on coding as ASCII text files. The kind of information a report should contain, e.g. codes, referenced coded text segments, the source text material, frequency of codes, etc., is controlled from the Analyse Codes window.

7.7 Discussion

HyperRESEARCH supports the creation and maintenance of codes and the on-line coding of texts based on the created coding list. It provides functionality for the search-and-retrieval of coding and generates reports with the results of the search. We found HyperRESEARCH easy to learn and use.

In comparison to other text analysis tools, the distinguishing feature of HyperRESEARCH is its functionality for formulating a series of rules and testing whether they hold or not with regards to the existing coding.

On the negative side, as far as the construction of a coding scheme is concerned, HyperRESEARCH does not offer the sophistication that programs such as ATLAS.ti, NUD*IST or WinMAXpro do. For example, HyperRESEARCH does not support the definition of relations between the created codes. Socio-demographic data or Case variables may neither be defined nor imported for analysis purposes.

HyperRESEARCH does not support memo-writing either for codes or coded text segments or the Cases themselves.

The organisation of analysis in Studies and Cases and how the Sources are related to them may prove confusing during the initial use of the program. HyperRESEARCH cannot access more than about 3,000 words at a time. If longer texts are to be analysed, the user must edit them by inserting a tilde character to indicate that a Source file comprises different 'pages'. One would expect that this task of separating long files into pages could be performed automatically by the program itself. HyperRESEARCH does not recognise or work with any other kind of structure within a Source text, i.e. paragraph, comments, different speakers of an interview, etc. This is restrictive for those analyses where information about different text elements of a Source text is required. Restrictive may also be the limitation on the amount of code references per Case, since this is not helpful when dense coding is required.

The Index Card window displays information for a *single* Case only and the user may request to see the next or previous Case or go to a specified one; it is not possible to obtain a display of all Cases with their respective codings and the codes they have been assigned to.

The functionality for text exploration (e.g. searches for words or word strings, KWIC display, word frequency of occurrence information, word index, etc.), is poor. The user may check whether a word or a string of words occurs in a Source text of a Case only in combination with the Auto Code option. The hyper-linking functionality is restricted to

the Index Card window: the user may go from the list of the coded text segments together with the codes they have been assigned to source text where the selected text segment is shown highlighted.

7.8 References and further information

The program is described in Depuis/Tornabene (1993) and in the on-line mini manual. The demo version comes with a mini manual containing a quick tour of the package. ResearchWare, Inc. maintains a HyperRESEARCH e-mail discussion list, i.e. hyperres-l, focusing on the use of HyperRESEARCH in qualitative data analysis.

Weitzman/Miles (1995), and Kelle (1995) review the earlier version (1.55) of HyperRESEARCH. More information about HyperRESEARCH can be obtained from: http://www.researchware.com.

The new HyperRESEARCH version 2.0 is soon to be released and has been announced that it will support digital image, audio, and video files next to text files.

8. KEDS (Kansas Event Data System)

KEDS is a program running on Macintoshes which has been specifically developed for the fully-automated or computer-assisted coding of international event data as these occur in English lead sentences of wire service reports (Reuters news service) or in chronologies.

KEDS uses pattern recognition methods for coding relevant parts of a lead sentence. Three types of information, each of which is stored as a separate file, are used for coding:

- proper nouns identifying political actors,
- verbs which delimit the most important part of the sentence determining the type of event – represented by an event code - to be assigned, and
- 'phrases' which help distinguish the different senses of a verb.

KEDS uses sparse and shallow parsing rather than performing a full syntactical analysis of the sentences. The accuracy and robustness of KEDS in analysing English sentences relies on exploiting text type and domain specific information of the source texts: sentences found in English newswire reports have a relatively consistent structure.

KEDS' development occurred around 1990 at the university of Kansas[11]. This review is based on the freely-available version 1.0 of the program.

8.1 Terminology

Composite pattern: a composite pattern contains a set of alternative matches.
Source: Refers to the actor of an event data record who initiated an action.
Sparse sentence parsing: Refers to the analysis of particular parts of a sentence, e.g. identification of verbs, their subjects and objects, but not full-blown and detailed syntactic analysis.
Target: refers to the actor in an event data record who is the object of an action.

8.2 Text importing and structure

In order to start working with KEDS the user creates a Project, which includes the data files which are used for coding, e.g. the verbs and actors files, as well as a 'logbook' for the coding performed.

KEDS can analyse text only if the source text file is structured in the 'KEDS-format'. This means the text is in ASCII format, organised into one paragraph and meta-textual information (such as the source of the text or date), precedes the actual news text.

---

[11] KEDS was developed by Philip Schrodt and his colleagues and can be downloaded from: http://www.ukansas.edu/~keds/software.html.

KEDS comes with additional reformatting programs for converting texts which have been downloaded from the NEXIS data service (a data service of Mead Data Central) to the KEDS-format. If the texts to be analysed are not in exactly the same format as NEXIS texts, the user must either edit the original texts manually or needs to write her own filter for reformatting the texts or modify the NEXIS filter provided.

The user may edit the texts while using the program after these have been imported in KEDS. The modified texts are then directly used by KEDS. Furthermore, the files with the information about the kinds of events, actors, and classes may also be edited during analysis. Therefore, KEDS enables the maintenance of the dictionaries used and text-update during analysis.

Output of the analysis are event data which may be edited on the screen before they are written to a file.

8.3 User interface

The KEDS working environment is organised in two main windows, namely the Text and Events windows. The former displays the text before it is analysed and the latter displays the parsed output. Only one news text at a time is displayed on both windows.

The possible operations in KEDS are mainly menu-driven. The software does not use icon buttons or drop-down menus or drag-and-drop operations. Exception to this is the point-and-click operation upon selection of a parsed sentence in the Events window. This brings up a new window with the event record which can be edited in the Text window. The files containing the Actors and Verbs as well as the source text itself may be opened and edited from KEDS, in which case additional windows are presented. On the whole, the user interface is simple, friendly, and functional for performing the kind of analysis KEDS was developed for. As a result, the user can learn fairly fast how to use the program.

The format or style in which the coded events should be displayed in the Event window as well as the format for saving the event data to file may be set by the user. The output variables, such as Event code, Source, Target, etc., and the their respective order are contained in a separate file, called Options. If no formats are specified then the default display and output ones are used: the display default format prints the following sequence on a separate line:

```
Source - Target - Event Code - Event Type – Matched Phrase.
```

Whereas the default format for saving the event data is:

```
Date – Source - Target - Event (code) – Event Code Text.
```

Furthermore, the KEDS project has developed a separate program, called KEDS_Display, for the graphical presentation in series, tables, and graphs of the aggregated event data of pairs of actors.

Rather than including a separate Help menu in the menu-bar for on-line help purposes, a help option for each single menu for a block of operations of the menu-bar is provided. On-line help is kept short and to the point.

8.4 Text coding and categorisation

KEDS uses extensive dictionaries of proper nouns and verbs as well as verb phrases to code sources, targets, and events in the lead sentences of a news text. The Actors and Verbs dictionaries, which contain the codes to be assigned and are used for parsing the texts, must be created before running KEDS. Both files have the same format, namely:

```
phrase [code] ;comments.
```

Codes may be less than or equal to 12 characters and the dictionaries entries are all spelled in upper case.

A list of proper nouns, which denote the actor who initiates an event or is the object of an event action together with the respective code for each actor, comprises the Actors dictionary. Actors may be countries, cities, persons, organisations, etc. For instance, the following are all actors included in the KEDS Actors dictionary:

```
DAMASCUS [SYR],
COMMON_MARKET [EEC],
CROWN_PRINCE_ABDALLAH [SAU],
HAMAS [PALHM].
```

The Verbs dictionary comprises simple verbs as well as verb patterns. As an example, consider the dictionary entry for the verb ‘to persuade’ which consists of a simple verb and its code, e.g. PERSUADE [031]. In contrast, the dictionary entry for the verb ‘to boycott’ comprises a simple verb and its code, but it also contains two verb patterns which are defined by a different code, i.e. BOYCOTT [191] - SAID * [172] - * WORK [181]. Note that the categorisation of both the Actors and Verbs dictionaries is not hierarchically organised.

To categorise and code relevant events, KEDS makes use of event coding schemes such as the WEIS scheme (World Events Interaction Survey, see McClelland 1976), the alternative dictionaries for the BCOW coding scheme (Behavioral Correlates of War; see Leng 1987) and the PANDA project at Harvard (Protocol for the Analysis of Nonviolent Direct

Action, see http://data.fas.harvard.edu/cfia/pnscs/panda.htm), which uses a superset of the WEIS scheme and has done extensive development using their own coding scheme.

KEDS uses the specifications contained in two additional files, namely Classes and Rules, in order to handle, on the one hand, by means of the former the variations, that is synonyms and different writings of a word or a phrase, such as 'USA', 'U.S.', 'White_House' all denoting the concept 'USA'. On the other hand, the Rules file contains re-write rules for treating two different structures as one, as for instance the transformation of passive voice structures into active voice ones.

The user can develop her own dictionaries or modify the ones delivered with KEDS and will probably need to fine-tune the dictionary for her analysis purposes.

Stemming can be used to match the different forms of the words. Special editing is necessary to prevent a character string from being used as a stem. A word followed by the underscore character will match only if it is followed by a space. If no underscore is included then a string of characters is interpreted as a potential stem.

KEDS codes either automatically or computer-assisted. During computer-assisted coding, the system pauses after analysing each text so that the user may confirm the suggested coding or modify it by editing the dictionaries. In the event that the text files with the verbs, actors, classes, etc. are edited, the parsing of events is updated while running the program. Fully-automated coding becomes advantageous when the user is fairly confident of the coverage and correctness of the dictionaries. The Auto-coding option may be used to code *only a portion* as opposed to all texts automatically and then be switched to machine-assisted coding mode.

KEDS uses a pattern language to define specific patterns for parsing typical structures of English news leads texts. Most of the codes used in KEDS dictionaries are simple codes assigned by the event coding scheme to actors and events. However, KEDS has three special codes, namely the 'null', 'discard', and 'complex' codes, which can be attached to phrases and change how KEDS deals with a sentence.

By default, pattern matching skips over intermediate words: for example, the pattern:

```
PROMISED
- * DOLLARS
```

will match the phrases 'promised two million dollars' and 'promised a grant of 10 million dollars'.

Also by default, phrase matching stops when a conjunction (e.g. 'and_', 'but_', specified in the Options or Class text files), occurs, so that a verb phrase cannot match words that occur after a conjunction. This does not hold for the Actors phrases.

KEDS supports the definition of negated composite patterns: a composite pattern contains a set of alternative matches and the negation of a set of matches, that is anything except the match defined in the negated pattern.

Since the aim of parsing and coding is to generate unambiguous event data for the lead sentence of a news message, overlapping or nested coding is not supported.

The output of the coding contains the variables which are set in the Options file. These control the formatting of the output, e.g. Date, Source, Target, Event, Coder ID and data, etc., and their respective values. For example, the following event data is in the default output format: `010106 CAL ITH 05 PROMISE`.

8.5 Text and coding exploration

The result of the analysis of each text is displayed in KEDS on a separate window. The user can set the display of the result with the respective coding so that only the text the user is working with each time is shown.

The user may ask for lists of all codes either for the one text analysed at a time or for all texts. This is helpful for developing and fine-tuning the dictionary entries while analysing.

Search operations in KEDS are possible only as far as the coding is concerned: the user can search and retrieve all codes assigned to a particular text. It is not, however, possible to search the text or all texts for particular words, strings or words or patterns. Indeed, general text exploration operations are not supported by KEDS. There is no word list that can be generated for a body of texts and there is no KWIC or Code-In-Context display. For example, it is not possible to have a display of all texts which were coded according to a particular verb code, and thus event or Actor.

The only frequency of occurrence information provided by KEDS is the frequency of codes which have been assigned during a particular stage of analysis, that is an analysis session. It is not possible to obtain frequencies of occurrence for all the words of a text – or for a news lead.

8.6 Export formats and interoperability

The default output which KEDS generates is standard event data of the form:

```
<date><source code><target code><event code><event code text>
```

However, the user may customise the output format by editing the Options file.

Because event data are an irregular, nominal-measure (categorical) time series, they must be aggregated before they can be used by standard statistical programs such as SPSS or graphical displays such as spreadsheets, which expect a regular numerical time series. The KEDS project has developed an aggregation program, called KEDS_Count, which automates the process of transforming standard event data. In order to use this program and develop the required command file, the user must be familiar with the Goldstein scales for the WEIS events data (Goldstein 1992).

8.7 Discussion

KEDS is a text type specific and domain- and language-dependent program which fully exploits this dependency. It has been tailored for analysing particular type of texts in order to output from text event data which may be further processed statistically.

KEDS has a restricted, but nevertheless very interesting approach to computer-assisted analysis of events data. It is optimised for the coding of event data and its accuracy is, naturally, dependent on the complexity of the source text, the event coding scheme as well as the kind of event being coded. KEDS seems to perform fairly well for the purpose it has been developed (see Gerner/Schrodt 1996, Schrodt/Gerner 1994, Gerner et al. 1994). It would be interesting to explore how possible it is to apply this approach and use KEDS for analysing the same kinds of texts in other languages. There has been work on developing a grammar for German in the past; however, the grammar has not been maintained in later stages.

In fact, one could imagine a modular design of a tool like KEDS where different grammar modules could be used or where one could switch between news leads parsing or full report parsing. For purposes of further development and expanding the scope of application of KEDS with regards to other languages, it would be worthwhile to investigate whether KEDS could offer documented interfaces to language dependent modules, based on which interested researchers could integrate own routines (e.g. as DLLs) for the language specific and grammar dependent parts of the program.

One of the shortcomings of the restricted support for text exploration KEDS provides is that the development and maintenance of the ‘dictionaries’ on which coding is based can become cumbersome and more laborious. Especially for the kind of text analysis which

KEDS supports, where a formal description is required of how specific words or phrases are used within the context of newswire reporting of political events, a richer text exploration functionality would be of great help. Having said that, there are a couple of facilities which may be used for text exploration purposes. More particularly, the separate utility program Actor_Filter may be used for searching a large set of texts for possible political actors (that is, proper nouns) that are not in an existing Actors dictionary. These are presented in a KWIC format in decreasing order of frequency. The Actor_Filter utility program "was used for developing a series of new dictionaries for various areas of the world (Russia, China, Colombia, Mexico, Albania, etc.), and was very helpful in rapidly developing dictionaries"[12].

Furthermore, if the user wishes to see all texts containing a particular set of phrases, she can use the complexity filter which will find the relevant texts and write them to a file. Although the original intention of that part of the program has been to define conditions under which a sentence is considered too complex to code, the user of KEDS may use it for that purpose. In other words, there are ways to perform some text exploration tasks with KEDS if one is already using it for other reasons, though the way these tasks are performed might not be as convenient as in other programs.

8.8 References and further information

KEDS comes with a detailed, approximately 200 page user's manual (Schrodt 1998) which, apart from being a guide to using the program, contains an explanatory paper on analysing events data sets and contains an extensive bibliography on KEDS and event analysis.

A short description of KEDS (with technical details) is provided in Schrodt et al. (1994). A statistical comparison of KEDS data with a human-coded data set is provided in Schrodt/Gerner 1994.

The KEDS project maintains a Web site at http://www.ukans.edu/~keds/. This contains the most recent versions of the program and the manual to download, as well as data sets and utility programs, a FAQ list, and copies of papers from the project.

---

12 Philip Schrodt, January 1999, personal communication.

9. NUD*IST 4 (Non-numerical Unstructured Data Indexing Searching and Theorizing)

NUD*IST is a text analysis software running under Windows and Macintosh. It was initially developed by Tom and Lyn Richards and later by Qualitative Solutions and Research Pty. It is supplied by Scolari, Sage Publications Software and by QSR.

NUD*IST can be used for the analysis of interviews, open-ended answers to surveys, group discussions, historical, legal documents, etc., as well as of non-text material, such as photos, maps, videotapes (with the CVideo software for Macs).

The user of NUD*IST can store, manage, code texts, search and retrieve both text and codes, write memos on documents and codes, and generate reports on the documents analysed. NUD*IST supports the intellectual coding of text segments as well as documents according to a coding scheme whose codes are held at nodes and may be organised hierarchically. Additional codes, which do not belong to the hierarchy, may be created and used for coding. The software provides basic text search and retrieval operations which can be combined with an elaborate search and retrieval system in order to explore the existing coding.

The merging of two or more NUD*IST projects and also the exchange of projects between the Windows and Macintosh platforms are both supported by the NUD*IST utility program Merge.

For our review we have used the demo version of NUD*IST revision 4[13], which is organised in four phases in order to introduce gradually the various features of the system. The only restriction of the demo version is that new projects cannot be saved.

9.1 Terminology

Command File: Similar to scripts or macros, a command file can automate almost any aspect of the NUD*IST work.
External Document: Any document which can be coded without being imported.
Free Node: A category which has not been organised in the Index Tree.
Index Tree: A hierarchical categorisation scheme.
Nodes: Primarily categories. As stated in the NUD*IST Handbook nodes are “containers for your thinking about the project and for your index references (coding) into the data document” (Richards 1998: 16). Nodes must have a title and an address and can contain a

[13] A demo version of the software can be downloaded from the QSR Web site: http://www.qsr.com.au/Software/N4/qsrn4.htm.

definition, coding or memo. Note, however, the meaning of the word 'ideas' is stretched: on the one hand, nodes *do* represent ideas for analysis but, on the other hand, they can be variables for the sub-grouping of documents, for example 'female', 'under 30'.
Pattern search: This term is used for any search for a pattern of characters, rather than a simple string. E.g. to specify an (unordered) set of words, e.g. [air | you | healthy] . The string of characters specified may include special character options.
Text Units: The smallest units of the document the user may code or retrieve, e.g. a page, a paragraph, a sentence, etc.

9.2 Text importing and structure

To start working with NUD*IST the user creates a NUD*IST Project and imports raw ASCII or ANSI files, called documents. Together with the imported files, external documents may be assigned to a project and used during analysis. The difference between importing a document or assigning it as an external document to a project is that although the user can code both, the external document cannot be browsed or the text searched. New documents may be appended to ones which have already been imported.

Not only texts but also tables of socio-demographic data for each case or document may be imported in NUD*IST. This operation automatically creates a new set of Nodes in the Index Tree and codes the texts used for the analysis. For example, the user can import case data with information about the respondents' sex, age or nationality, and NUD*IST will create new nodes respective of the variables and code the texts accordingly

Imported text files can be edited with the program's own text editor and the system updates subsequently the existing files. Thus whilst editing, adding or removing text units rather than any characters, NUD*IST allows the combination of editing and coding. Editing in NUD*IST does not invalidate subsequent coding.

The raw ASCII files can contain some minimal structure which consists of headers and sub-headers. The concrete text structure need not be determined by the user prior to importing the texts, but pre-processing allows automatic coding. Sub-headers and headers may be inserted and changed after the document has been imported NUD*IST. The structure allowed is top-down with the header containing information about the project, the speakers of an interview, etc., and the sub-header dividing the documents into sections (with each section delimited by a * character) and the text unit for the text to be coded. The text units are delimited by marking them in the text with hard carriage returns, which is normally the case for delimiting paragraphs. If the user wishes a text unit other than paragraph, the text has to be either pre-processed or edited in NUD*IST. Carriage returns must be inserted after a sentence, a line or even a word. Of course, in the latter case the result

would be that text is difficult to browse when coding is performed. NUD*IST automatically numbers the text units and can generate a report with the relevant numbers or with statistics about proportions of documents retrieved.

9.3 User interface

NUD*IST is graphics-based. It uses multiple windows, the operations on which are partly possible by click-on buttons and which contain some pop-up menus.

The NUD*IST working environment consists mainly of two windows: the Document Explorer for handling the documents for text analysis and the Node Explorer (or Index System) made up of the nodes, i.e. categories which the user has defined or the system has automatically created if a table with socio-demographic data has been imported.

The Document Explorer lists the documents of a project. Upon selection of an item from the list, NUD*IST displays the two properties of a document, namely its name and its header. These properties can be edited. The user can browse a document by selecting it in the Document Explorer (Browse button) or write a memo on the document (Make Memo button) or obtain information about the date the document was last modified (Properties button). The option for browsing a document will bring up two windows: the Document Browser, whose header shows the name of the document displayed with the text, and the Palette window, which contains a list of buttons for editing text and coding as well as for examining the coding of a document. Clicking on a text unit updates the Document Browser header with the respective number of this text unit.

The Node Explorer shown in figure 11 displays the types of nodes which can be created in NUD*IST:

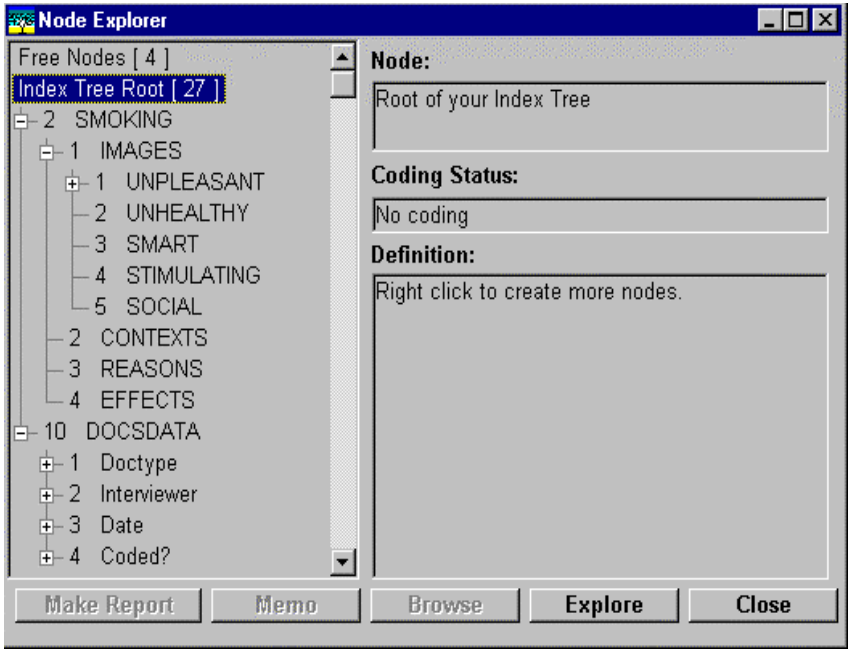
1. Free nodes
2. nodes which have been ordered in an Index Tree
3. nodes created during text searches
4. the node for Document Annotations, i.e. comments inserted a document, and
5. the Node Clipboard.

Some operations on the Node Explorer, like creating shifting, copying, attaching, merging or deleting nodes, are possible by selecting node and bringing up context-sensitive pop-up menus.

Any node may be created from the menu bar or from the Node Explorer or during coding. Nodes organised in trees may also be created from the Index Tree Display, a graphical tree presentation. This contains a Tree Summary which shows which sub-part of the tree is displayed each time and shows always only two levels of the hierarchy. If the user wants more levels, she can duplicate the window. Note that both windows are live and can be at different places in the

Index System. Up and down arrows may be used to move to the higher or lower level, but only two levels are shown each time. All levels can be viewed in the Explorer. Tree diagrams may be printed or saved as Decision Explorer (http://www.banxia.com/demain.html) or Inspiration formatted files (http://www.inspiration.com).

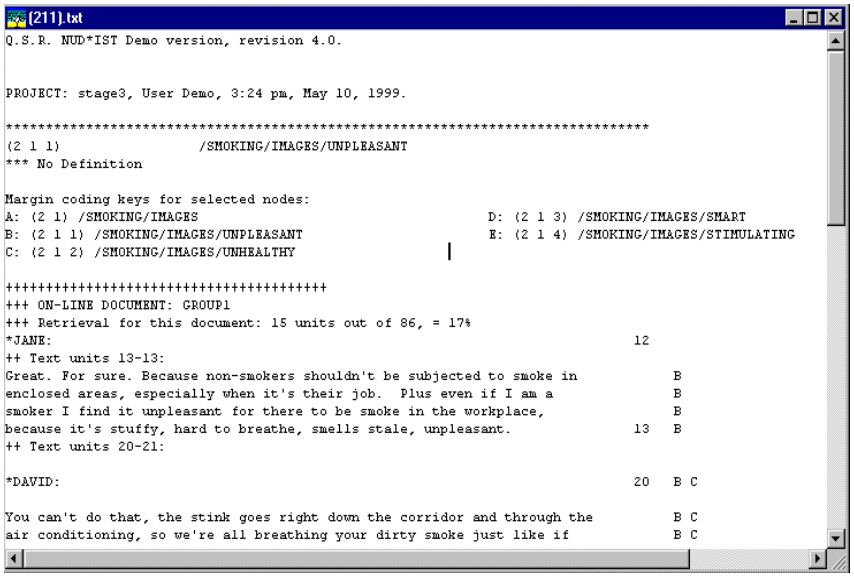
Figure 11



By hitting the Browse button the user may choose to go from a node (only one node may be selected at a time) to the text coded at the selected node at any time. From the text coded at a node, the user can use shortcut keys to select the displayed text units and either jump to the original document or examine the coding of the displayed text. The latter option provides a list of all other nodes to the selected text. The user may select a node and browse the text units assigned to it.

The reports which NUD*IST generates on a node are displayed in a separate window. The coded text units are shown with the codes at the end of each unit or in the margin as in the example of a generated report shown in figure 12.

Figure 12



A summary with some general text statistic information about the number of text units or percentage of retrieval in comparison with the total amount of text units is included in the end of a report. The coding display options in NUD*IST include reports with coding "stripes" in the margin; a coding summary that provides text unit numbers coded at each node or an awkward end of line code display.

The two possible views of a text, either as raw text or as coded text, are controlled from the two NUD*IST windows by means of browsing a text or generating a report on it. The consequence of this is that the user is kept 'away' from the actual text. During the browsing of a text coded at a selected node, the available operations do not include text exploration next to coding ones. In particular, when browsing a text coded at a selected node it is not possible to search for a word or a string of words within the displayed text. This makes working and exploring texts pretty cumbersome. Although the user can locate a text unit by number in the Browser or search text in a report as well as use the text search facility to search for a word or pattern, she cannot search text in the window displaying a text coded at a selected node.

Text search in NUD*IST is menu-based: the user selects the respective menu option, types in the word to be searched, and initiates the search.

The user may automate almost any aspect of the NUD*IST work by means of creating Command Files. These are particularly handy for complex processes, such as complex search requests, which the user might want to use frequently. This functionality may help the user to automate boring clerical tasks during analysis.

The on-line help of NUD*IST is comprehensive and can be searched as table of contents or by providing a keyword.

9.4 Text coding and categorisation

Coding is based on nodes created by the user. Each node must have a name and an 'address' and may optionally contain a definition, coding (i.e. information about the text units it has been assigned to), and a memo. By address is meant that the node has a numerical value in the coding hierarchy, i.e. the Index Tree. All nodes are displayed in the Node Explorer. Hierarchic relations between nodes may be defined in NUD*IST. No other types of relations, e.g. cause, part-of, or sibling, are supported. The Index System has a top-down structure; nevertheless, NUD*IST supports the creation of free nodes which are not organised in a hierarchy and which the user may in a later analysis stage index. Nodes can be deleted, or moved from one Index Tree node to another one as sub-nodes or may be merged to another node.

Coding in NUD*IST means placing a reference to one or a series of text units of a document at a node, for example the text units 24 to 29 are referenced at both Free Node *x* and Index Tree node *y*, and the whole document 'Group1' is referenced at the Free Node *x*. Coding is 'bi-directional': the user can select either a single text unit or a series of text units to be coded (on the Document Browser) and assign it to a node or can 'tell' the system which text unit(s) should be coded, e.g. all questions. If the node the user wants to code at does not exist, the system asks whether it should create it. In view of this, coding in NUD*IST is also one way of creating nodes. Alternatively, new nodes may be created on the Node Explorer or the menu bar on the Index Tree node. Wherever the node is created, the user tells where it should be placed by selecting from live Explorer display that allows shifting the Index System as wished.

NUD*IST does not enable the user to select only a part of a text unit as opposed to the whole text unit and code it (as in ATLAS.ti). Only a single or a series of text units may be coded. A unit may be assigned to any number of nodes.

Automatic coding is supported by NUD*IST in the following way: the user can specify a Pattern Search (see Terminology section), create a node for the search output, specify the amount of context to be coded, and then code all texts or documents or sub-parts assigning them to this node. To a certain extent this operation parallels the dictionary-based coding of content analysis programs.

The Node Browser permits viewing and re-coding of coded text, changing of nodes, and creation of new nodes. Coded segments can be re-coded, the coding of a segment can be examined and the user can jump to the source to view it in context to do more coding. A useful feature with regards to coding is the Spread-Unspread option, whereby existing coding can be interactively spread in order to view a wider context and code it at this or another.

The user may create, view, edit or delete a memo attached to either a document or a node. It is not, however, possible to select a text position or text unit and create a memo for it. NUD*IST offers the option to "annotate" a specific text position or text unit, that is to simply select it and include it in a list of Document Annotations, which is a node of the Index System. Inserted annotations can be coded, searched, and retrieved according to author or topic using index searches.

NUD*IST 4 comes with QSR Merge, which is a utility program for users of NUD*IST who want to merge two or more projects and/or to translate projects between Macintosh and PC platforms. In that way NUD*IST supports teamwork and provides multi-platform functionality.

9.5 Text and coding exploration

A report on a selected node may be displayed (and saved) with a summary of all text units coded with the selected node plus some general descriptive statistics included.

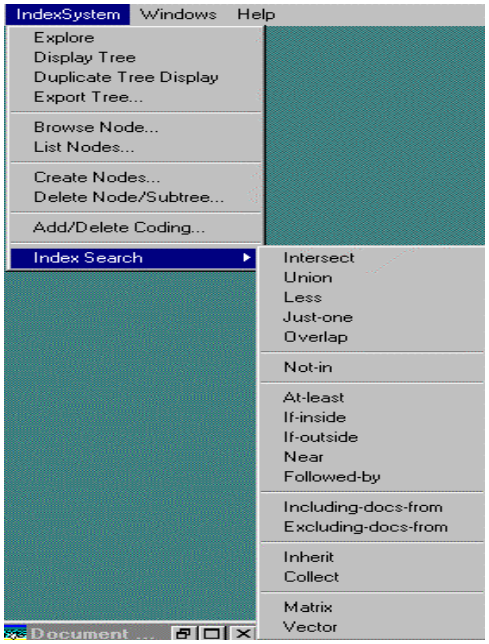
Searching in NUD*IST is distinguished between text search and index search. The retrieved instances for both types of search are 'coded' at a node. Regarding text search two types may be distinguished: string search, otherwise called pattern search, and special characters search. Pattern searches are used for looking for a list of words as a set of alternations. The result of every search is displayed on a separate window with the search string shown in upper-case characters so that it is distinguished from the rest of the text. NUD*IST makes a new node for the retrieved positions in the Text Searches area of the Node Explorer. This way the user may browse the result similar to browsing a node of the Index System.

The string search can be narrowed down so that only particular contexts are searched for occurrences of the defined string. For example, only texts which have been coded according to a specific node, or documents coded at a specified node, should be considered.

Furthermore, the search can be narrowed down to consider the search string as single tokens, or in particular locations, using wild cards. The user can search for a string under a certain category. This type of search is interactive, whereby the user can include or exclude particular hits.

Pattern search enables the user to define an unordered set of strings of characters, e.g. `[air | you | smoking]` or a set of words having a common stem, e.g. `lo [ving | ver | ved | ves | ve]`, and look for all occurrences of this set in the texts. Some restricted use of wild card characters is allowed for pattern search.

Figure 13



The Index Search option offers seventeen different operators as shown in figure 13 for searching the Index System and performing analyses of 'patterns of coding' at nodes.
The advantage of index searches is that they allow one to seek relationships or test hypotheses about the relationships between categories and patterns of one's coding.

There are index search operators, for instance At-least or Near, which can be used for contextual searches: these look for text segments which have been coded with a given code,

based on the node's contextual relation to text segments coded with another code. In addition, contextual searches can be used to look for possible co-occurrences of two nodes within a user-determined distance. Another form of index searches are Collation Searches, e.g. specifying the various types of co-occurrence of codes on a text segment. Special operators use the possibility of 'trees' of nodes; 'collect' makes a node with all the coding of subcategories of a given nodes. 'Matrix' and 'vector' allow qualitative cross-tabulations of one group of subcategories by another.

The user may filter index searches for only certain types of documents or sections coded at any node. Since nodes can be used to express values of variables, such as 'female' or 'male', this is interesting in that it enables the user to search on dynamically produced 'sub- corpora'.

NUD*IST makes a new node with the result of a search, attaching it to the Text Searches area of nodes and creating a kind of 'search library'. The user may edit and/or save the result of the search under this node and browse it again at a later stage, or add it to the Index Tree and, thus, use it for coding. Results of any text or index search may be used as input to further index system searches, and in this way one can build up complex searches.

NUD*IST provides no word list or index, comprising all words of the project's documents which could be used for searching for certain words or strings. In order to search for a word string, the user needs to know beforehand if this word occurs at all. Word search can be fairly tedious and, as already mentioned, on-text selection with the mouse for searching purposes (either for a word or a string of words) is not supported; instead the user must choose the Search menu option and use the respective window each single time, even for simple searches. It is possible to define a search which contains a combination of pattern or text search *and* index search in either of two ways. The text search can be restricted to text or documents coded at a node (and that node can be made from any combination of nodes, via index search). Thus, the user can retrieve all instances of the word 'smoking' in those text units coded with the node 'UNHEALTHY'. Or the nodes from text search or index search results can be combined in other index searches. Thus, the user could ask for all occurrences of 'smoking' if they occur after 'lung cancer', and in the same document there is coding at the node 'UNHEALTHY'.

9.6 Export formats and interoperability

For the graphic display of the relation between document and category or text unit and category, NUD*IST exports coding in four types of tables: Document Base Data tables, Case Base Data tables, Coding tables, and Matrix and Vector tables.

A coding table contains a summary of a particular part of the overall coding performed in NUD*IST and may be input into a statistics package or spreadsheet. The user must provide in the Export window the address (e.g. number) of a node and a table is saved as a matrix with headers. This, of course, makes it necessary to pre-process the saved file if the user wishes to import into a statistics package such as SPSS or SAS. The table shows for the specified node whether or not it has been assigned to each of the documents. If the specified node contains sub-nodes, only one sub-level is taken into consideration. Note that the user may export only the coding according to *one* node each time, which makes the export of the whole coding to statistics very work-intensive.

Furthermore, NUD*IST exports part or all of the Index Tree together with nominated details for further processing by Inspiration or Decision Explorer.

9.7 Discussion

The user of NUD*IST can develop and maintain a categorisation scheme according to which a number of texts can be coded on-line. The automatic creation of a new set of nodes is supported if these are organised as tables of socio-demographic data for each case or document. Tables of such data may be imported and the documents of a project are coded accordingly. Nodes, thus, can be *heterogeneous* entities as their purpose is to code a document as a whole or text unit of a document.

NUD*IST supports further the theory building process by providing functionality for exploring the categorisation scheme, as for instance the powerful Index Search option which supports the searching for patterns of coding. Creation of memos for either documents or nodes is supported as well as the generation of reports containing the text and its coding.

Regarding the NUD*IST working environment a drawback of the program is that the user is kept away from her text. A considerable improvement in working with NUD*IST would be a third window, let us call it Text Explorer, on which not only coding could be performed, but also searches could be initiated and the contents of which would be updated every time a new node or document would be selected.

The program is graphics-based and uses windows technology, but the graphical means, i.e. font type, use of colour, etc., for displaying a text or the search results are fairly primitive for modern software.

The NUD*IST text exploration functionality is restrictive: Neither a word list of all word types of a project (or document) nor KWIC displays are possible in NUD*IST. Such information would help the user to define more targeted text searches for a word or a string of words. Search definitions which include combinations of word strings and codes

are supported and can be automated or rerun via command files. The generated reports with the retrieved occurrences for a search include some descriptive statistics, but apart from that general frequency of occurrence information is not provided in NUD*IST.

The readability and the general presentation of the coding or search results could be improved by using colour or different font types. Moreover, the tree display, which is a helpful means of visualising the categorisation scheme, is too narrow and confining as only a very small portion of Index System may be displayed at a time. Incorporation of information visualisation algorithms and modern ways for displaying such information would improve tree displays.

The user cannot view the whole text/document with the whole coding. It is possible to obtain a display of all coded instances of all text with both coded and non-coded parts of it showing in 'coding stripes' for up to 26 nodes with code and text side by side. This is not available, as, for example, in ATLAS.ti for the browsed text whilst coding.

NUD*IST supports the export of an existing Index System to re-use in another analysis project in either of two ways: a duplicate project can be emptied of documents, and therefore of coding, or an Index System can be recreated via command file. It is not possible to import in NUD*IST a non-NUD*IST categorisation scheme (functioning as an Index System) to use for analysis.

With regards to the categorisation scheme, NUD*IST optionally organises nodes hierarchically, i.e. parents or children of a node. However, to use other relations, e.g. *cause*, *part_of* or *sibling*, export to other software is required. Reliance on prior tree structures can support a top-down approach to analysis. Flexibility of the Index System and ability to code on in the interactive Node Browser are used more with the grounded theory approach.

9.8 References and further information

The demo version is comprehensive and user-friendly. Thanks to an on-line tutorial guide which is organised in four stages, it is well-structured and easy to understand and follow. The full version comes with a 88 pages User's Guide (Richards 1998), which provides also advice on research design and details on text search and its uses. Qualitative Solutions & Research Pty Ltd (QSR) offers a variety teaching material for NUD*IST. A guide with exercises for NUD*IST is available (Bazeley 1998) as well as (Microsoft) PowerPoint presentations to demonstrate NUD*IST. All this material can be downloaded from the QSR Web site http://www.qsr.com.au.

QSR produces a NUD*IST newsletter and maintains a mailing list for NUD*IST users. It also organises both introductory and advanced users workshops world-wide (see Web site).

Reviews of earlier versions (N3) of the software are provided by Weitzman/Miles (1995), Kelle (1995), Kelle (1996), and Walsh/Lavalli (1997). Barry (1998) reviews NUD*IST (comparing it with ATLAS.ti). A long literature list on NUD*IST, with reports on using the software and technical papers is available at http://www.qsr.com.au/events/referenc.htm.

QSR Merge comes with a 52-page manual that describes installation, use, and how to plan a project merge. It also comes with both Macintosh and Windows QSR Merge diskettes to facilitate cross-platform merging.

10. QED

QED is a Windows program for qualitative data analysis distributed by MiMiC (Holland)[14]. The software supports storing of text and coding, searching and retrieving texts as set of texts, creating codes, on-line coding of text, and searching for codes.

All sorts of qualitative data, such as interviews, survey data, observation data, archive material, protocols or transcribed conversations can be analysed using QED.

We reviewed the QED demo version 1.1007. The demo is restricted in that it allows only ten Data Cards (see next section) to be displayed and searched.

10.1 Terminology

Analysis Card: similar to a memo, it may be used to keep track of ideas and observations.
Data Card: texts in QED are divided into 'meaningful' chunks and placed on Data Cards.
Set of Cards: A cross-section of the cards in the data files which "answer to a set of characteristics", and which are included in a Workbook. The user specifies what these characteristics are. Sets can be build automatically and manually.
Workbook: A system file which provides the framework for a QED analysis project. If the user wants a data file to be included in the Set operations this has to be included in the Workbook. In other words, Sets and Set operations are only possible for Workbooks.

10.2 Importing text and structure

The user can import to QED ASCII text files structured in a 'QED-specific' way. Imported texts are stored on the different Data Cards and each Data Card contains a user-defined 'meaningful' text unit. While the user can split up the data in small units, placed on cards, larger subsets of cards can be organised in different files. Individual or groups of files can then be analysed in different Workbooks. Alternatively, texts can be typed directly onto the Data Cards in QED.

Up to three different variables may be manually assigned to each Data Card. Any of these variables may be used for text filtering purposes, e.g. for considering only texts with a specific variable value.

Texts contained in the Data Cards may be edited, new cards can be added to an existing number of cards or cards can be deleted while working with QED. QED does not support the import of a separate text file containing a list of codes to be used for coding.

---

14 TriMM MultiMedia, Holland designed and implemented QED.

10.3 User interface

The QED working environment provides menus with shortcuts only for the editing menu options, icon buttons, and point-and-click operations. QED uses the Data Card metaphor for organising and displaying the texts, the output of a search, and even the on-line help topics themselves. The program produces an index of all texts, which in fact is a list of all Data Cards of the opened Workbook, and the user may point-and-click on a Data Card. This provides a separate window with the text of the selected Data Card and a list of the codes which have been assigned to it. The search output provides an index of all Data Cards containing an occurrence of what has been searched and the on-line help index the help topics or titles as Data Cards. Point-and-click is also used for displaying a Data Card from the on-line help or search indexes.

Another way of browsing a Data Card is by using backwards and forwards arrows provided in the text display window. These take the user to the previous or next card in a predefined Set as opposed to the whole collections of cards belonging to an opened Workbook. Cards can be manually 'linked' to each other, a feature which allows the user control moving from one card to another.

If there are any variables defined for the Data Cards, their values are included in the Data Card browse window. A check box indicates whether the card is included in the Active Set of Cards. If a code in the code box is selected and the code has been assigned to a text segment (as opposed to the whole Data Card) the coded text is highlighted.

The Code List window lists all codes assigned to texts. The code list is automatically created by the program when coding is performed and the user may not edit it. If a text displayed on a Data or an Analysis or a Help Card is printed, only the part of the card on the screen is printed instead the complete text of the Data Card. The coding performed as well as search output can be printed as a report including text, codes, variables, and analysis cards.

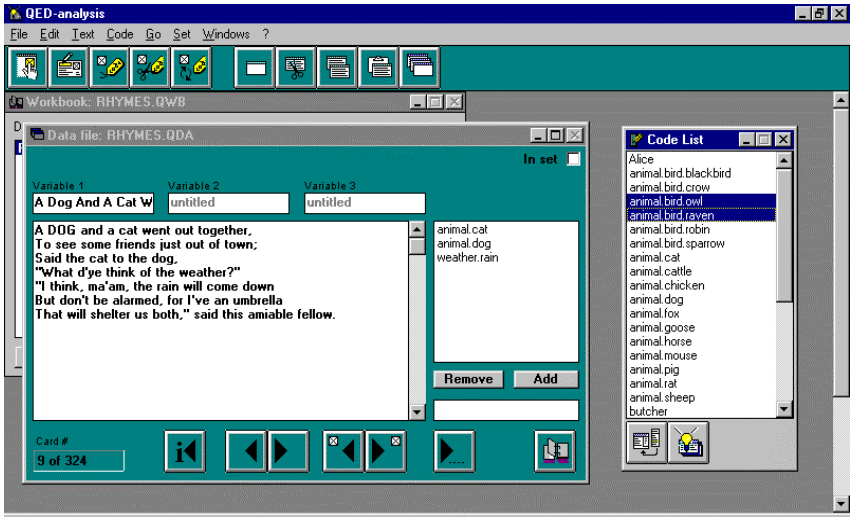
10.4 Text coding and categorisation

Apart from the three variables which may be assigned to each Data Card by the user, coding in QED means that the user assigns codes to either a Data Card or a text segment of a Data Card. For example, in figure 14 the three codes listed at the right hand side of the Data File window have been assigned to the text card displayed. The three codes have been selected from the Code List window.

If no text selection is made on a Data Card, the code is assigned to the whole card. If a text segment is selected, the code is assigned only to the selected text segment. The user either types the new code in the list displaying the codes assigned to a particular Data Card or

selects it (or a number of them) from the Code List window and assigns it (or them) to the Data Card. New codes can be created by directly typing them in code box of the text display window. The code list is automatically updated when a new code is created. QED does not support the definition of relations, e.g. hierarchy, part of, etc., between categories: the code list consists of a number of unrelated categories.

Figure 14



QED supports nested coding, where parts of an already coded text segment can be assigned to different codes. However, as far as the display of nested coding is concerned QED displays only the text segment which was coded last. The same text segment may be assigned to more than one code. Furthermore, codes can be assigned to groups of texts resulting from searches. For example, the category 'animal' can be assigned to all texts in which the words 'dog' or 'sheep' occur or that have been already assigned the category 'bird'.

The QED user can create memos and link them to a text card or to a category. The categories contain the date of creation and may be modified but cannot be searched for.

10.5 Text and coding exploration

QED supports the user to perform simple searches for a word or string of words but not case-sensitively. Nor does it support searches for a code, or a variable value in an active text. Wild cards are allowed in the search process.

Furthermore, the user can search all Data Cards by means of defining a Set. The result of a Set definition is a Set of Cards, which is a group of Data Cards that have been filtered according to up to four user-specified criteria. Since QED marks the search results by checking each Data Card as belonging to, the user may jump from one card in a Set to the next card. The defined searches can be stored as different Sets.

The search may both contain and combine codes, variables, Data Card IDs or text strings. Query operators, such as *exact match (=)*, *contains ({})*, *bigger than (>)*, *smaller than (<)*, does *not equal (<>)*, may be used. In addition, the four criteria can be combined with the logical operators AND, OR, and NOT. Codes or new variable values may be assigned to the whole Set of Cards. Moreover, the user may remove or change a specific code assigned to a card in the Set or delete or export cards in a Set.

10.6 Export formats and interoperability

Apart from saving the texts, subsets of texts (defined by Sets) with the variable values and codes as ASCII files, no other operation or export possibility is supported by QED.

10.7 Discussion

QED appears to have been designed similar to the model of 'traditional' qualitative analyst working without computer: one uses index cards and writes the text on these cards, sorts the cards by different criteria and selects groups of cards. Its functionality does not go much beyond this model. In that sense it is fairly limiting, especially compared with other qualitative data analysis programs, which support creation of code hierarchies or code networks and their visualisation, definition of relations between codes, some text statistics, searching memos for keywords, obtaining lists of all coded text segments, and different formats or exporting coding for further analysis.

In the coding process QED allows coding of text segments in Data Cards. Nevertheless, one cannot use these segments in the search because search is designed only as a card search.

We found the working environment of QED difficult to learn and get used to, and although it would have definitely helped to have the user's manual, we still believe that the graphical user interface is not self-guiding as claimed by the QED distributors.

10.8 References and further information

The full version of QED comes with a 50 page manual. More information about the program is provided at http://www.trimm.nl/qed/.

11. TATOE (Text Analysis Tool with Object Encoding)

TATOE is a text analysis program combining some functionality typical of qualitative and quantitative text analysis[15]. Although TATOE has been developed with the aim to support text analysis in general, that is independent of disciplinary context, it carries a bias towards linguistic analysis as well as text analysis for the humanities. This bias is, for instance, demonstrated by the importance placed on supporting the incorporation of linguistic annotation such as part of speech information. The available version runs under Windows 95/NT and Unix.

The program may used for the analysis of texts such as articles, public speeches, news messages, etc. as well as dialogues. TATOE combines interactive coding with text exploration functionality, e.g. word searches, word frequencies or concordances of selected words. Not only raw texts but also texts which are already coded may be imported in TATOE. The user can either import or create one or more coding schemes and use them for coding text; this way the user lays directly or indirectly by the text import operation different kinds of analysis information related to different levels of interpretation and text description. Searches on text as well as coding may be performed and the user can code the results of a search automatically. The text data, categorisation schemes and their categories as well as the styles which the user has created for the display of coding may be exported to XML. Furthermore, the coding data may be exported to SPSS for further processing.

We have used the freely-available beta version 0.9 (for Windows) of TATOE for this review.

11.1 Terminology

TATOE-Notebook: a separate window which is organised as a virtual 'notebook' whose 'pages' are dialogue windows for performing main operations in TATOE, e.g. creating a coding scheme, importing and exporting texts, etc.
Occurrence Lists: a TATOE-Notebook page which contains a logbook of searches which have already been performed and their results.
Search Pattern Definition: a user-defined search pattern which can contain words, strings or categories or a combination of them, e.g. look for all occurrences of the word 'party' followed optionally by either the category 'Values' or the category 'Political Systems'. Search pattern definitions may be stored as rules of a grammar which may be re-used for searching or coding purposes or be included in other search pattern definitions.

---

[15] TATOE's development was started by Melina Alexa together with Lothar Rostek from GMD-IPSI, Darmstadt, while Melina Alexa was employed there.

Textbase: an object network or a kind of database produced by importing the texts.

11.2 Text importing and structure

TATOE does not work directly with a text file; rather the user must import the text into a textbase. Along with the text material, a textbase contains pointers to the positions of all the words of a text corpus, all the coding schemes - if the text corpus analysed contains coding - and their respective categories as well as the coded text segments.

Texts already imported into a textbase are considered as 'static' text data which are not to be edited during analysis.

To create a textbase the user goes first to the Textbase page of the TATOE-Notebook: she provides a name for a textbase, specifies the directory it will reside and TATOE creates a textbase.

The import process may start after a textbase has been created. It comprises three steps, namely

- specifying the textbase into which the texts are going to be imported
- providing the name(s) of the file(s) of the text material
- specifying the format of the texts.

TATOE supports the import of XML, HTML or plain ASCII text files. Furthermore, if the text data to be analysed are dialogues (in ASCII format) the user can select the dialogue format option and type in the delimiters used for the dialogue turns or speakers. The XML import is only possible for text which has been exported from TATOE to XML. In other words, XML is used by TATOE as the internal exchange format for text data.

In addition, TATOE supports the import of German texts which have been linguistically analysed with the morphological analysis program Morphy (see Lezius et al. (1998) and http://www-psycho.uni-paderborn.de/lezius/). The way this works at the moment is rather long-winded: the user must first import the texts into TATOE, then save each text and analyse it separately with Morphy, and then import the morphologically analysed text into the textbase.

TATOE supports the import of new text data into an already existing textbase

TATOE distinguishes two text structural levels: the first level, called Paragraph level, is the coding level, that is a word or a string of words (up to a paragraph) may be selected only within a paragraph to be coded. Selections over this level for coding purposes are not possible. For dialogues, the paragraph corresponds to a dialogue turn. The second structural level, called Text level, is for organising and displaying the texts of a text corpus.

The total of texts to be analysed are organised in a text corpus.

In order to analyse a text corpus the user must first activate its respective textbase.

The import of a coding scheme independently of the import of text data is supported: the user creates a new coding scheme and types in the name of the ASCII file which contains a list of the scheme's categories.

11.3 User Interface

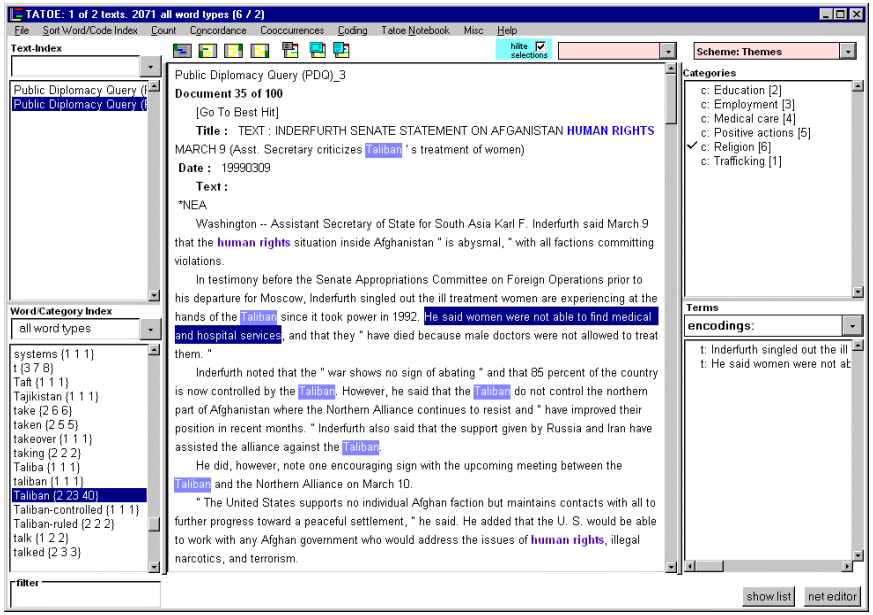
TATOE has a visual interface keeping the text data, the coding, and the categorisation scheme(s) close to each other and supporting their 'interaction'. Similar to ATLAS.ti, TATOE makes use of object-oriented design, and effort is made to keep the necessary operations close to the data they are applied to. The TATOE desktop is divided in five panes, which are ordered underneath the menu bar and the button bar (see Fig. 15). The latter contains buttons for performing frequent operations, such as switching between a concordance or a full text display or obtaining a concordance or a KWIC display.

The five panes are:

1. *Text-Index*: displays a list of all texts of an activated textbase
2. *Word/Category Index*: it displays
   - either a list of all words of the activated textbase together with their frequency of occurrence according to the whole corpus, the total of texts each word occurs in and the total of paragraphs
   - or a list of all the categories of each categorisation scheme. Each category is displayed together with frequency of occurrence information, that is how many times altogether, in how many texts and how many paragraphs each category has been assigned to.
3. *Text pane*: one or more texts are displayed in this pane in full text mode. The same pane is used for concordance displays.
4. *Categories* pane: displays all the categories of a categorisation scheme sorted in alphabetical order
5. *Codings* pane: displays the list of coded text segments. There is a number of display modes for this pane and the user selects the one she requires in the Codings pull-down menu:
   - *All terms*: by choosing this setting TATOE displays only those coded text segments which contain the word or string of words the user has previously selected, i.e. all coded text segments which contain the selected word with all the different assigned categories.

- *Encodings*: with this display mode the user gets a display of coded text segments which a selected text position has. If the user moves the cursor anywhere on the text pane, then the TATOE updates the display and the coded text segments change according to the selected word or string of words on the text pane.
- *Marked Instances*: in this mode the Codings pane displays all *new* coded text segments, i.e. all the words or strings of words the user has coded since the beginning of an analysis session with TATOE. This mode is the default mode and is set when one starts working with TATOE.
- *Terms*: in this mode TATOE displays in the Codings pane only the selected word or string of words with all its coding variants.

Figure 15



Upon selection of one or more texts in the Text-Index the texts are immediately displayed on the text pane. Coded text segments can be displayed in colour if the user has defined one or more display styles. In the Styles page of the TATOE-Notebook the user can create her own style for displaying coding in colour by specifying which colour should represent which category. A number of categories of one or more schemes may be selected and assigned to

particular colours. These selections can be stored under a display style name and be used during analysis. The user may create as many styles as she finds necessary for her work and during analysis may select one of them for displaying the text in the text pane.

Apart from displaying the coding on the text pane the user may set the display mode of the Codings pane to Encodings in order to always get feedback about other existing coding of a text segment at the text pane on which the cursor is placed.

If a word is selected from the Word Index, TATOE updates the Text-Index with only those texts in which the word occurs and displays in the text pane the first text in which the selected word occurs. The word is highlighted in colour. The user may select all texts or only some from the Text-Index to be displayed in the text pane.

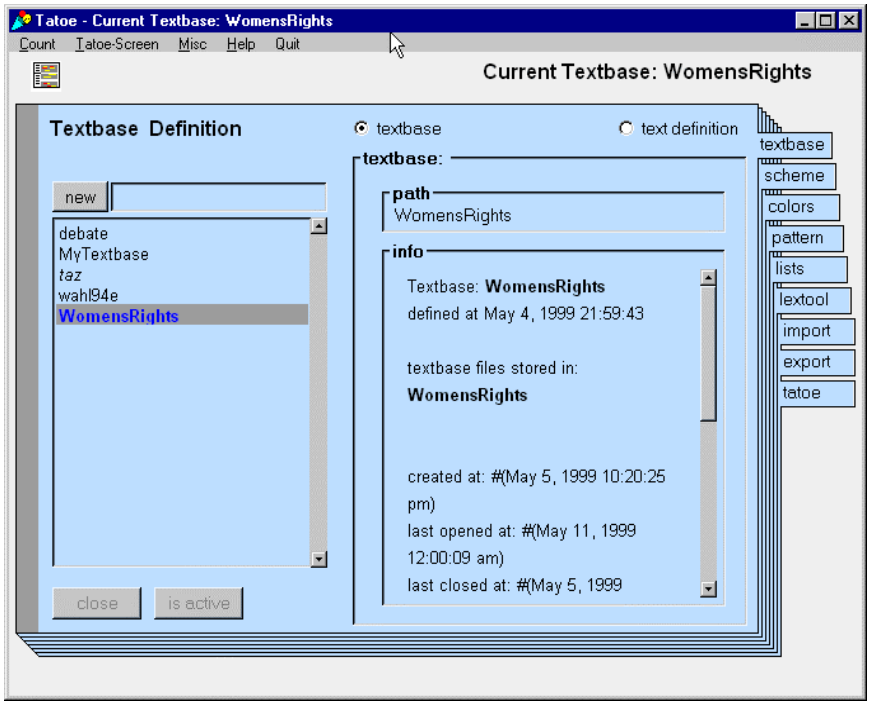
Selected words, strings of words, scheme categories, and coded text segments may be displayed in the concordance mode where the occurrences of the selected items are shown with their left and right context. Two kinds of word-in-context displays are possible: a typical KWIC display, with one line for each occurrence and the selected token shown in the centre of each line in bold characters or the concordance display where the amount of left and right context for each occurrence is not fixed; instead the user determines the amount of left and right context to be displayed on-the-fly by moving a slider. Each occurrence of the selected item(s) is highlighted colour in the concordance display. Furthermore, each occurrence is not shown in a single line, but rather as separate text chunk, similar to the paragraphs of a text. Note that the user may select a word or a string of words from the concordance display and ask for a concordance for the new selection.

The user opens the Coding Screen window to code. This displays (in way similar to the Windows NT/95 Explorer) all existing schemes and their respective categories. Schemes can be ‘closed’ or ‘opened’, in which case all the categories of a scheme are listed below it. The user may move categories within one scheme and thus change the hierarchy, but cannot move one category from one scheme to another. If a word or a sequence of words is selected on the text pane, those categories which have already been assigned to the selected segment are crossed.

To code, the user selects a word or a string of words on the text pane and then selects a category from the Coding Screen. In addition, the user may code all the ‘key-words’ or occurrences of a concordance display in one go by assigning them to one or more categories.

The TATOE-Notebook is an essential ‘tool’ of the program. As shown in figure 16 this is a virtual notebook with page markers for separate dialogue interfaces corresponding to the different operations that can be performed in TATOE.

Figure 16



In particular, each Notebook page contains a separate dialogue interface for:

1. creating, selecting, and activating a textbase
2. importing texts
3. creating and maintaining categorisation schemes
4. creating and storing display styles for the coding
5. defining search patterns
6. storing the already performed searches and their results, called Occurrence Lists
7. exporting texts, their coding, schemes or styles.

The Notebook communicates directly with the TATOE main desktop. The user may ask for a concordance from the TATOE-Notebook and this will be displayed in the text pane of the TATOE desktop.

11.4 Text coding and Categorisation

In order to code in TATOE, the user creates a categorisation scheme. The scheme is identified by unique name and contains an unlimited number of categories. A category must belong to one scheme only, has a name and an optional numeric value. Categories may not be moved from one scheme to another: they can be created, deleted or re-named only within a single Scheme.

The program supports either flat or hierarchic categorisation organisations and no other relations can be defined to hold between categories. Category hierarchies are created in the Coding Screen window.

TATOE supports the creation and maintenance of parallel coding schemes. In this manner different kinds of interpretation may be organised separately from each other but at the same time they are available for coding as well as text and coding exploration purposes. For example, linguistic coding may be organised as a separate scheme from interpretative coding. During coding the different kinds of information organised in separate schemes may be used for search and retrieve operations.

As already mentioned, morphologically analysed German texts with the Morphy tagger may be imported in TATOE and be further analysed. In that way additional coding may be used for supporting further analysis.

The smallest text unit that may be coded in TATOE is the word. The largest text unit which may be selected for coding in TATOE must not be over a 'TATOE paragraph' (see section 11.2). Manual on-line coding in TATOE follows after selecting a word or a string of words and assigning it to a category listed on the Coding Screen window. The user has the choice to code either the selected string only, or all identical occurrences of it in one step, the latter being some kind of automated coding. The same selected string may be assigned to more than one category and a part of an already coded text segment may be selected and assigned to a different categories. In other words multiple, nested, and overlapping coding is supported.

New text data may be imported in an existing TATOE textbase. If the text data are same as those already in the textbase with some additional coding or if they are only partially the same TATOE, cannot support their comparison and the subsequent merging of information which is not redundant. In that sense, the support of the program for collaborative work is fairly restricted.

TATOE does not support memo-writing.

11.5 Text and coding exploration

TATOE creates an index of all the word types of a textbase. Each word type is displayed in the Word Index pane together with the frequency of occurrence. Three different frequencies of occurrence are provided: how many times a word occurs in the whole textbase, in how many texts and in how many paragraphs altogether. Selecting a word provides immediately the first text in which the selected word occurs. Only one word may be selected from the Word Index pane at a time.

All categories of each categorisation scheme are displayed in the Category Index pane together with frequency information as to how many text segments have been coded according to each one of them. Upon selection of a category, TATOE displays the first text to which this category has been assigned and highlights the particular text position in the text. Furthermore, a list of all words or strings of words assigned to a selected category may be generated and displayed on the Category Index pane.

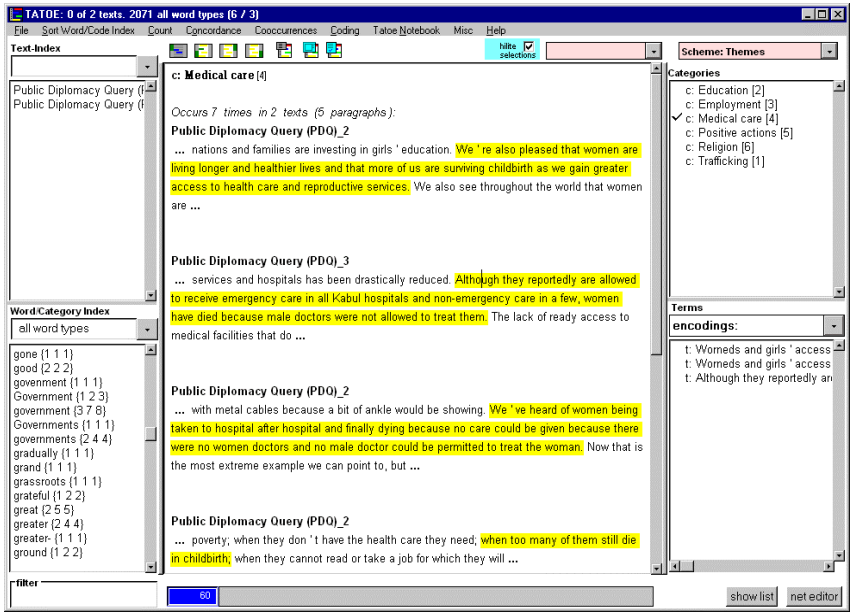
At any time during analysis, the user can explore either text or coding switching from one type of search to the other. Selected words or strings of words can be seen either as they occur in a single text or in the concordance or KWIC displays containing the total of all their occurrences. TATOE also provides concordance displays for both scheme categories and coded text segments. The user may select more than one category or coded segments as well as a combination of both categories and coded text segments and ask for a concordance display of them. For example, in figure 17 a concordance for all the text segments assigned to the scheme category ‘Medical care’ is provided.

Searches with wild characters can be defined in the Word/Category Index pane: the user types in the search string and the result is a filtered word list displayed on the pane of all words matching the given string.

More advanced searches are possible in the search pattern definition page of the TATOE Notebook. A search pattern may contain words or word strings as well as scheme categories. The definition of search patterns in TATOE may be seen as a kind of grammar development. The user can save each search pattern as a rule under a specific name. This rule may be used recursively within other patterns, therefore also rules.

Two kinds of searches can be initiated for a given search pattern: either for those occurrences which match the defined pattern or for those which do not. The result of a search is displayed as a concordance display and the system provides feedback about the frequency of occurrence for each element of the pattern as well as the whole. The matches or non-matches for a search pattern may be coded in one step.

Figure 17



Frequency distribution graphs for the categories of a categorisation scheme are provided by TATOE. The user can choose between a distribution graph based either on the total of tokens or types of the words coded.

TATOE stores all performed searches as Occurrence Lists which may be viewed and operated on from the Occurrence Lists page of the TATOE-Notebook. The user may either select an Occurrence List and initiate it again to view its retrieved text segments. Moreover, Lists may be merged or subtracted in order to obtain combined results, i.e. the addition or the difference of two lists.

Collocation information is provided in TATOE, by means of the Contexts menu option. This generates for a selected word or string of words a list of all phrases which occur either to the left or to the right of the selected string together with their frequency of occurrence.

11.6 Export formats and interoperability

It is possible to export the coding from TATOE into the SPSS statistics software. The program generates an SPSS syntax file, which can be directly used with SPSS. The text

unit over which information about the codes is provided is a paragraph. All categories are shown with labels.

Furthermore, TATOE supports the export of texts, categorisation schemes and their categories, coding and display into XML.

The coding schemes can also be saved as ASCII files.

11.7 Discussion

TATOE is a text analysis program which provides functionality for organising the coding into a number of parallel flat or hierarchically organised coding schemes. The coding of text is supported either in a manual or semi-automated mode. The latter is possible in connection with a definition of a search pattern. TATOE will export texts and coding to XML as well as the export of the coding data in a format for statistical processing with SPSS.

The user interface of TATOE uses windows technology with a variety of pull-down and pop-up menus, visual means such as colour, icons, etc., and provides different views of the analysis data, i.e., different concordance displays, frequency distributions graphs, graphs of categorisation schemes, etc. There is a high degree of interlinking between the different displays and some of the TATOE windows.

The possibility to keep coding in separate schemes and at the same time be able to combine categories from different schemes for search-and-retrieve operations or coding, is advantageous both for text exploration and further coding.

Categorisation schemes may be both imported and exported in TATOE. That way coding schemes which have been used for the analysis of other texts can be imported and re-used.

For those interested in exporting their texts and coding to the XML format TATOE is advantageous. However, more work needs to be done in TATOE to fully support this.

Some of the features of TATOE which are either restrictive or not supported enough concern:

- advanced search-and-retrieve techniques for the exploration of coding
- the size of the unit which may be coded.

With regards to the first aspect, compared with such advanced tools as the ATLAS.ti QueryTool, or the Logical engine of WinMAXpro for querying the coding and hypothesis testing, TATOE is fairly restricted. Although the program supports searches which combine word strings with categories, its functionality for defining queries with a variety of semantic operators is not rich enough.

A major restriction of TATOE, especially for analysts who code large text segments, is the fact that coding is limited within the paragraph level. A further restriction of TATOE concerns that it provides no memo writing functionality.

Although the user can specify which structural unit belongs to the paragraph level, for example, for the analysis of short answers to open-ended questions each answer text may be a TATOE paragraph, the mechanisms of TATOE for working with different text structure information are limited.

TATOE provides no functionality for incorporating in the analysis additional kinds of information, such as socio-demographic data.

11.8 References and further information

TATOE is described in Alexa/Rostek (1996) and Rostek/Alexa (1998). An example of using TATOE for linguistic text analysis is described in Alexa (1998) and the application of TATOE for the analysis of German news messages is described in Rostek/Alexa (1998) and Rostek (1999). The program is freely available and can be downloaded from http://www.darmstadt.gmd.de/~rostek/tatoe.htm. A short tutorial may also be downloaded from the same site.

12. TEXTPACK

TEXTPACK is a Windows 95/NT program for computer-assisted content analysis. It was developed and is distributed by ZUMA.

TEXTPACK allows the user to code a text corpus according to user-defined dictionaries. It supports text exploration by providing word frequencies, cross- reference lists, KWIC displays, comparisons of the vocabulary of two files, and the development and validation of the dictionary used for coding. TEXTPACK exports different kinds of coding results, for example, frequencies of categories, to statistical packages like SPSS or SAS for further quantitative or logical analyses.

TEXTPACK can be used for the analysis of any type of text, such as open-ended questions, interviews, newspaper articles, etc., and it can efficiently handle large amounts of text.

We reviewed the user version of TEXTPACK for this study.

12.1 Terminology

GO Words: Words which are considered for the analysis.
STOP Word List: Words excluded from further analysis.
Words, Multiple-word Combinations, and *Word Roots*: A word in TEXTPACK is a string of characters which is preceded and followed by one (or more) blanks. A Word Root is defined as the left part of a word. Multiple-word Combinations are defined as sequences of single words.
Identifications (IDs): Markers for each text unit in a text corpus.

12.2 Text importing and structure

TEXTPACK accepts raw ASCII text files, which may contain identifiers (IDs) denoting text units, e.g. a paragraph or an answer to an open-ended question, or meta-textual information, e.g. speaker, text header. The texts can be structured hierarchically and maximum three 'levels' for IDs may be specified, whereby every level represents some external feature of the text.

Texts cannot be edited within TEXTPACK, but the dictionaries used for coding and the GO/STOP word lists can.

12.3 User interface

The TEXTPACK desktop is menu based with a number of menus for file processing, text exploration, coding, and filtering texts according to IDs.

The imported text is displayed when the Display Text option of the TEXTPACK file menu is

used. This Text Display window must be closed in order to use the other menus. Thus, the user is kept away from the text analysed and after having worked for some time with TEXTPACK viewing the text becomes fairly cumbersome.

Selecting a menu option in TEXTPACK brings up a menu window where the user sets a number of parameters depending on the particular option, e.g. the sort order of the frequency list or the type of coding. The result is then displayed in a text window and may be printed or saved as file.

A status bar in TEXTPACK displays which text and dictionary are used and whether a filter is set.

TEXTPACK is not interactive in the sense that it is not possible to select on-text and perform a particular operation. Furthermore, the different TEXTPACK windows do not communicate with each other.

Finally, TEXTPACK provides detailed on-line help, as a separate Help menu in the menu-bar, organised in a hypertext structure. In addition, a help option for each single menu window is provided, which is kept short and to the point.

12.4 Text coding and categorisation

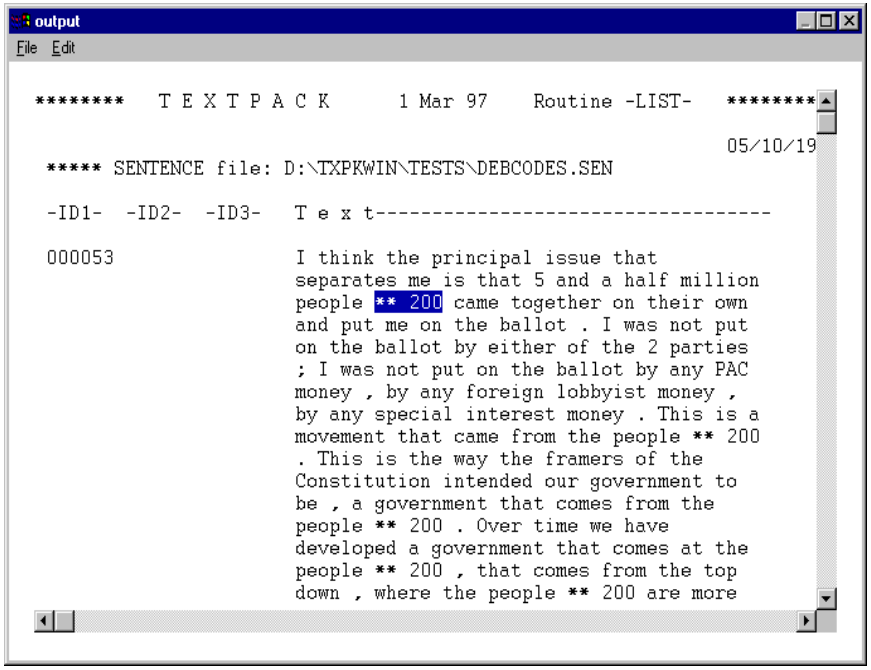
Coding in TEXTPACK means that a word or a string of words are assigned a code based on a dictionary. The user has already constructed the dictionary and tells the program where it is stored. The dictionary entries can be single words, or word roots or strings of words. Coding in TEXTPACK is case sensitive.

TEXTPACK codes a text corpus automatically. The user cannot select a word or a string of words on-screen and assign it to a code manually.

Multiple coding of a text segment is supported, whereby the user specifies whether multiple or single coding should be performed. If single coding has been set, the strings of words as opposed to single words have priority. For example, if the dictionary used contains the entry ‘American’ with the code 201 and a separate entry for ‘American people’ with the code 200 and the user has chosen multiple coding, whenever an occurrence of ‘American people’ is found in text both codes (200, 201) are assigned to it. Otherwise, code 200 for ‘American people’ is given.

The result of coding is either a file containing the frequency of categories for each ID or a file containing all category numbers in the order of their occurrence in text for each ID or a file containing the coded text. For the latter, each coded word is followed by the number of the code assigned as shown in the example provided in figure 18.

Figure 18



If a string of words has been coded, the code follows the last word of the string. Special characters may be used for denoting the code numbers, e.g. a star or a hash, so that the codes can be distinguished from text.

TEXTPACK supports dictionary validation, for example, by producing a list of all dictionary entries which have been used for coding, listed according to each coded ID, or a file with all IDs which do not contain any coding.

The program does not support the writing of memos on either text or categories or coded text segments.

12.5 Text and coding exploration

TEXTPACK generates a list of all word types of a text corpus with the text reference of each word type. This list may include the GO words or may exclude the STOP words.

In addition, TEXTPACK will generate a word list with the frequency of occurrence for each word. Different sorting options, e.g. alphabetically or according to increasing or decreasing frequency of occurrence or according to word endings, are possible. Furthermore, the user may set the word list to include

- either all word types or
- only those words which occur more than *n* times. The user may, for example, exclude from the word list all words which less than three times.

With TEXTPACK word comparison lists can be produced for two different text files according to the following ways:

- A comparison word list may contain all words of both texts, for each word its frequency of occurrence in both files is provided as well as the difference of its frequency.
- Alternatively, the user may obtain a comparison list which contains only the new word types occurring in the second text file with their frequency of occurrence.

Finally, TEXTPACK provides KWIC displays, sorted alphabetically or by IDs or the assigned categories for a number of words. The user has at her disposal a number of possibilities for determining the amount of context displayed. For example, the context may be displayed in one line either without any IDs or with IDs. In addition, the context of the key words may be displayed either as a single text unit or as a group of text units with the unit immediately before and after the one containing the key word also shown. The amount of context shown in a KWIC display may not be flexibly determined by the user.

TEXTPACK does not support on-text selection for overall text searches for a word or a string of words.

The user can filter the amount of texts displayed as well as the amount of text which should be considered for coding. This filtering is possible either by specifying the text unit IDs or by providing, for example, socio-demographic variables. That way TEXTPACK supports the generation of sub-corpora.

12.6 Export formats and interoperability

TEXTPACK exports the results of coding in an SPSS or SAS syntax and raw data file. The results of coding are exported as frequencies of occurrence for each category. Furthermore, TEXTPACK may write to a file the sequence in which the codes occur, e.g. Code1 followed by Code7 for the ID1. In addition, the coded text can be saved in an ASCII file.

12.7 Discussion

TEXTPACK is a content analysis program for the automatic coding of texts based on a dictionary provided by the user. The coding can be exported in the appropriate format for further statistical analysis with, for example SPSS and SAS. The development as well as validation of the dictionary are supported by means of a number of functions for text exploration.

The functionality of the program is aimed mainly at supporting the tasks of dictionary development and automatic coding. Further statistical analysis must be performed outside TEXTPACK.

Although TEXTPACK offers a relatively wide spectrum of functionality for text exploration, the support of the software for the exploration of the coding performed is restricted. The program's purpose is to code text automatically and export coding for further processing with statistics software. In a way, it is assumed that any 'exploration' regarding coding will take place during the statistical processing.

The main limitation of TEXTPACK, with regards to coding, is that although the dictionaries may include not only words, but also strings of words, the coding process is more or less single word based and context independent. Furthermore, there is only one mode of coding possible, namely automated, and no interactivity is provided for accepting or correcting coding suggestions as, for instance, supported in KEDS.

A final point concerns TEXTPACK's desktop, which does not make full use of sophisticated GUI technology and is relatively rigid.

12.8 References and further information

The full version of TEXTPACK comes with a user manual (Mohler/Zuell 1998). Text analysis examples applying TEXTPACK are given in Mohler/Zuell (in press) or in Zuell/Mohler (1992). Further references can be found on http://www.zuma-mannheim.de/research/methods/en/textanalysis/publik.htm. A yearly workshop is organised which provides an introduction to computer-assisted content analysis using TEXTPACK. More information about TEXTPACK (including the downloadable demo) can be found at the following Web site: http://www.zuma-mannheim.de/software/en/textpack/.

13. TextSmart

TextSmart is a software for coding answers to open-ended survey questions. It runs under Windows 95 and NT and has been developed and distributed by SPSS Inc.

TextSmart uses linguistics technology, such as word stemming, and statistical algorithms, e.g. clustering and multidimensional scaling, for generating automatically 'categories' for the coding of survey responses. TextSmart is advertised as "dictionary-free", in the sense that there is no need to create a coding scheme or 'concept dictionary' before running the analysis. The quality of the automatic coding depends on the quality of both the Exclusion and user-defined Alias Lists (see section 13.1). However, the user may also create her own categories simply by specifying which words or combinations of words belong to a category.

In both automatic or interactive creation of categories, codes are automatically assigned to the texts. Apart from automatically categorising the texts, each text can be manually assigned to one or more categories.

Tables, charts, and graphs enable the user to verify the categories. The assigned codes can be exported to SPSS or other statistical package for further analysis.

We used the full version of TextSmart 1.1 for our evaluation. This was kindly provided at no cost to us by SPSS, Inc.

13.1 Terminology

Alias List: Contains a list of related words which are all considered equivalent and should be thought of as groups of synonyms. Typically, two kinds of alias lists are built: semantically-related and morphologically-related (failed to be joined during stemming).
Brushing: A TextSmart feature which helps to investigate the relationships between words, categories, and responses. When brushing is 'on', each time the user clicks on a category or a text or an Included Term or Excluded Term the selection is broadcasted across all the other windows.
Excluded Terms: Words or aliases that have 'little semantic value' (as opposed to 'content words') and cannot be used to build categories.
Included Terms: Words or aliases that are used to build categories, either interactively by the user or automatically by TextSmart. The inclusion list is built by the program using all words in the text and the alias list without the terms in the excluded list.

13.2 Text importing and structure

TextSmart accepts tab-delimited ASCII text files of English answers to open-ended survey questions. The answers as well as the questions to which they were given, are structured according to the typical structure for this text type: Questions are composed of a question ID and the specific text of each question, whereas answers are composed of a questionnaire (or case) ID, a question ID and the actual text. Any supplementary information the user wishes to store to a survey text file is stored under a special field, labelled Survey Information.

Another alternative for data entry in TextSmart is to use the SPSS script "Export To Textsmart.sbs", which creates tab-delimited text from SPSS data files. This script is provided with TextSmart.

Text files which can be input to TextSmart may contain responses to more than one open-ended questions in a survey. In TextSmart the user selects which questions she wants to analyse at a time.

In addition, TextSmart allows you to match your texts to any SPSS data file that contains socio-demographic information.

It is possible to edit the text after it has been imported in TextSmart and the new text is processed after it has been saved, taking into account the new words for the automatic categorisation (for the included term list). Additionally, a spelling check option helps to correct the text data.

13.3 User interface

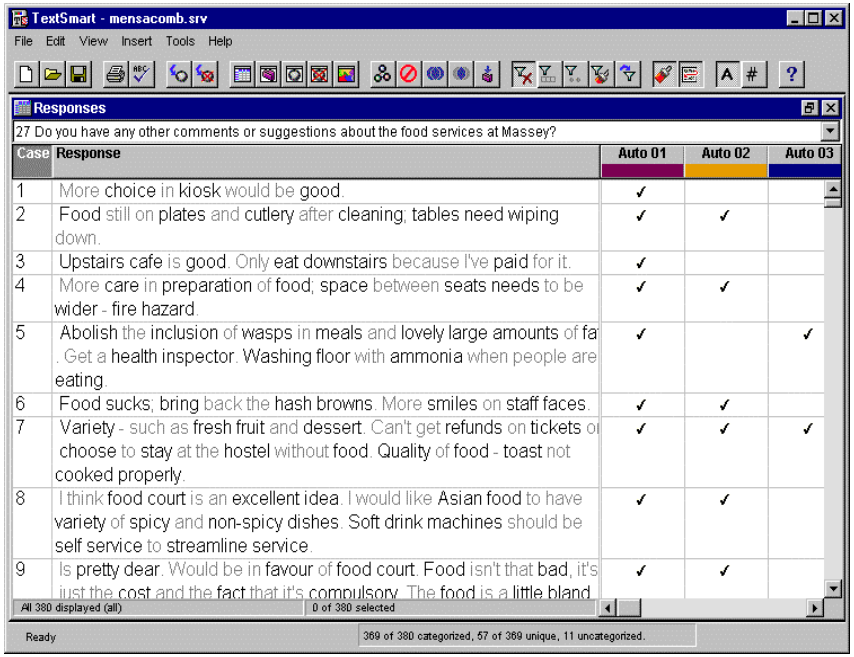
After the text files have been imported in TextSmart, the raw text is displayed in the Responses window in tabular form comprising both the associated questionnaire IDs and categories each text belongs to. This is shown in figure 19.

The user may select a survey question to work with and only those texts which are answers to the selected question are displayed. The Responses window shows the texts and all words contained in the exclusion list are greyed. On the right side of the text, all categories (if there are some defined) are displayed as 'checked' for each text. The texts can be sorted by categories, text ID or alphabetically according to the first word in each text. If a text is selected and a socio-demographic data file has been used, the demographic values can also be displayed for this text.

Both included and excluded term lists as well as the categories are optionally displayed in separate windows. There is, also, a separate window listing the categories. Words and aliases can be moved between both the excluded and included term lists as well as the list

of categories by using drag-and-drop. Both included list and excluded term lists can be sorted by frequency of occurrence of each word or alphabetically. Each word in both lists is followed by its frequency in the text. The categories are followed by two columns Count and Unique. The Count field displays the total number of responses for each category that fall within a category. The Unique column gives the number of responses that are categorised as unique to the selected category. The categories may be sorted by their category number or the total of texts coded according to each category. When a term is added to a category, TextSmart updates on-the-fly the coding: the Count and Unique fields are filled in, and the responses that fall into the category become selected in the Responses window.

Figure 19



The Word Frequencies Chart, Category Frequencies, and Categories Plot windows are generated automatically and are not interactive.

TextSmart offers an detailed on-line help based on search terms.

13.4 Text coding and categorisation

The excluded and included term lists (including the aliases) play an important role in the categorisation process. The included term list contains all words or aliases in the text except those words contained in the exclusion list. The excluded term list includes words or aliases which are considered as 'meaningless' for the coding (e.g. articles).

TextSmart offers a predefined excluded term list which may be changed by the use. The user, however, can also prepare a completely new list of excluded terms to work with. In both lists aliases can be defined which build groups of words thought of as synonyms.

TextSmart has an internal alias list for frequently used words which handles the word stemming. The program performs an automatic lemmatisation when the texts are first imported as well as every time the user manually updates the responses. Words having a common root (e.g. run, running, runs, ran) are grouped together. TextSmart creates an alias (a category) for these, identified by the root word and distinguishes them from other aliases by placing an ellipsis after the word. Lemmatisation is only available for survey responses in English. The user cannot change or export the internal alias list.

Additionally, TextSmart offers the possibility to define a user-defined alias list. During the automatic process of defining categories, all words combined to an alias are handled as one word. Therefore, the specification of an excluded term and alias list is very important. All words in the exclusion list are not considered for coding.

When preparing a text, it is important to remember that alias lists do not handle phrases. Therefore, for those cases where a string of words needs to be considered as one unit, hyphens have to be used to join the words.

With regards to the automatic coding functionality of TextSmart the user may code by:

- Term Frequency, which means that only the most frequent used words are used for coding or
- Clustering only (see below) or
- Clustering combined with Term Frequency.

The Clustering procedure produces categories based on word co-occurrences. This is a three-step process:

1. The program creates a matrix of similarities from the terms (words, aliases, and stems) in the included terms list. It pairs each term with every other term in the list and checks to see how often each pair occurs in a text (a response), that is, how often it co-occurs. It constructs a contingency table for each pair of terms in turn. It uses this information to compute a binary measure, the Jaccard similarity measure (Text-

Smart User's Guide, p. 45ff), for each pair of terms. The measure consists of the number of co-occurrences between two terms divided by the sum of co-occurrences plus non-co-occurrences.

2. In the second step, the program hierarchically clusters the similarity matrix and places the clusters into a user-specified maximum number of categories. The cluster algorithm used attempts to produce clusters whose largest distance between any two members is as small as possible; it tends to produce 'compact' clusters.
3. TextSmart displays clusters using multidimensional scaling in two dimensions to scale the matrix of similarities. This is useful, especially if only a small number of terms has been included for the analysis.

As far as the option for automatic coding where the Clustering method and the Term Frequency options are combined is concerned, TextSmart first builds the categories in the clustering process and then uses the most frequently used words from the remaining word list for the term frequency categories.

If the categories are built by automatic categorisation, a category plot is displayed with all words of the categories. The user can decide whether she wants to delete all previously automatic generated categories or if the new categories should be created in addition to previous generated categories. The automatic categorisation process allows a maximum of 25 categories in one run. The maximum number of terms on which categories are to be based is 300 terms.

The user can define her own categories, by using drag-and-drop from the included or the excluded term lists or another category into the user-defined category. After the concepts (i.e. terms) which belong to a created category have been defined, the user must choose between the Boolean operators OR, AND or NOT for relating the terms. Cumulative categories may be created by creating a new category and copying existing ones into it.

The coding assigns each survey response to one or more categories. Although the coding process is based on a word list, the result of the coding is the categorisation not of the respective word in a text, but rather of each response text according to the specified set of categories. This is the usual case in the content analysis of answers to open-ended questions. Regarding the display of coding, the program does not display which exact text segment has been assigned to which category.

There are two ways of categorising texts according to categories: Either by constructing a category and defining the criteria, that is, which words should be considered for categorising the texts as described above, or by manually assigning texts to categories. The latter categorisation makes the response an 'exception'. If the category's criteria change, this

text remains assigned to the category. Similar to manual assignment of a text to a category, a text may be removed from a category, which, in effect, causes the particular text to be excluded from the criteria of the category specified. If the category's criteria change, the text selected will *never* be assigned to that category.

Categories which were automatically or are user-defined cannot be re-used for another project, since they are based on the inclusion term list, which contains only the words of the selected texts. The alias and excluded term lists can be imported and exported and therefore re-used.

13.6 Text and coding exploration

Search in TextSmart is possible only through the included and excluded term lists. If one or more words or aliases in the included or exclude lists are selected all texts containing the words are displayed in colour. In the included and excluded lists multiple selection is possible. For example, the words 'cheap' and 'clean' have been selected in figure 20 and the those responses which contain the selected words are highlighted.

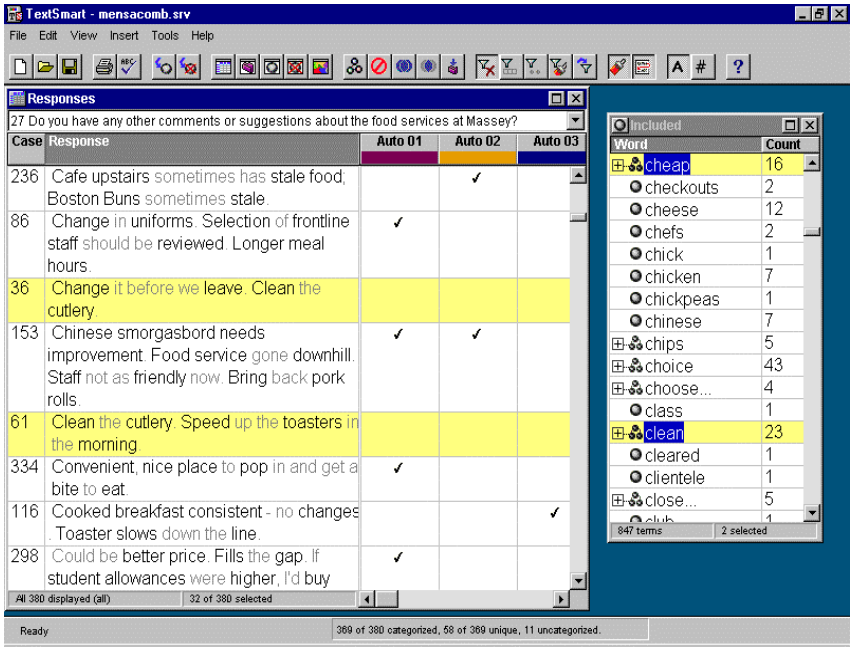
Selecting a category highlights in the Responses window the texts coded according to the selected word and highlights the category in the Category plot window. Only one selection is possible at a time.

The responses can be sorted by any of the categories in order to group all texts belonging to the same category together. This is helpful because the user may then check the categorisation and the definition of the different categories.

Three different windows presenting frequency of occurrence information of either the terms, i.e. Word Frequencies Bar Chart, or the categories, i.e. Category Chart. The third window called Categories Plot displays the clusters. Multidimensional scaling in two dimensions is used to scale the matrix of similarities. The Word Frequencies Bar Chart displays the 20 most frequent terms in a survey. This is a fairly small number and might, therefore, be a limitation for some analyses.

Three filter options may be used to reduce the amount of texts displayed at one time: (i) non-categorised responses, (ii) multiple categorised response or (iii) *brushed* responses. The latter helps to ensure that the categorised responses fit with the categories criteria. Alternatively, the user can filter those texts which have not been categorised yet and have only these displayed. This makes it easier to categorise them accordingly. Of course, the user may choose not to use any of these filters and have all texts displayed.

Figure 20



It is possible in TextSmart to import and have available socio-demographic data while analysing. If one selects a response in the text window the demographic information of the selected respondent will be displayed.

During analysis the user can print a report as a survey summary or a category summary or category charts or categorised responses text, but it is not possible to save these as separate files.

13.6 Export formats and interoperability

TextSmart exports the questionnaire ID, the question ID, and the assigned codes as an SPSS data file or saves them in a tab-delimited ASCII file. These files may be used in statistical programs for further analysis.

The exclusion and alias lists can also be saved as ASCII text files, but only those aliases created by the user and not those which were created by means of lemmatisation are

included. The texts may also be saved as ASCII text files, but with no information about the category membership of the texts.

13.7 Discussion

TextSmart has been especially developed for coding responses to open-ended questions in surveys and exporting the coding for further statistical processing. The functionality of the program is restricted to the category development, supporting automatic and manual category development, and to coding.

The building of categories is word-based and phrases are not considered. The coding is also word-based, whereby words can be combined by AND, OR or NOT operators. Because of this, TextSmart may be suitable for projects which do not require complex or sophisticated categorisations, such as keyword-like answers to open-ended questions.

TextSmart is language-dependent since the stemming process is language-dependent and, at the moment, stemming is possible only for English texts.

Our testing has shown that the results of the automatic categorisation are usually not good enough and can serve only as a possible basis for developing 'meaningful' categories. The amount of work required to change the generated categories or build new ones may be large; especially if one considers that good excluded term and alias lists are of primary importance for the coding. Therefore, one had better not take the TextSmart advertisement of a "dictionary-free" coding (see http://www.spss.com/software/textsmart/overview.htm) at face-value.

The user interface of TextSmart is clearly designed and easy to learn and use. However, the program itself is *very* slow and requires patience and time; for example, using drag-and-drop for moving words from one term list to another or to aliases or categories takes an awful lot of time. Finally, although the User's Guide is very helpful, we think that a tutorial would be beneficial.

13.8 References and further information

The TextSmart User's Guide (TextSmart 1997) not only explains how to use the program, but also describes the techniques used for, e.g., the automatic categorisation procedure. The last chapter of the User's Guide contains helpful tips for using the program as well as for the methodology the user adopts.

Further information about TextSmart can be found at http://www.spss.com/software/textsmart/ including a FAQ (frequently asked questions) list about the program.

14. WinMAX97Pro(fessional)

WinMAXpro is a Windows program for qualitative-oriented text analysis. It was developed by Udo Kuckartz in Berlin and is distributed by Sage Scolari Publications. The program is available either as a standard or as a professional Windows (95 and NT) version.[16]

WinMAXpro may, in principle, be used for the analysis of all types of texts appropriate for qualitative text analysis, such as interviews, case histories, field notes, letters, etc.

The program supports on-line coding, the construction and maintenance of a hierarchical coding scheme, memo writing, the definition and usage of case variables, and the exploration of coding by means of logical operators. It also calculates frequencies of occurrence of either words or strings of words or codes and supports teamwork. If there are variables attached to texts, WinMAXpro exports them together with their assigned values and respective texts in an appropriate format for further statistical processing. Furthermore, the coding can be exported for further statistical processing.

Our review is based on the free of charge, trial version of WinMAXpro 97. This version is the same as the full user version with the exception that the user cannot save her analysis work or export the coding.

14.1 Terminology

Activating Text: The user can control which coded text segments should be displayed by WinMAXpro. The user may activate one, more than one or all texts. Activating a text is necessary also for performing all operations related to a text, e.g. searching, obtaining frequency of occurrence information, etc.
De-activated Text: The coded text segments of a de-activated text are not included in the window containing the list of coded text segments. One, several or all texts, as well as one or more codes must be activated/de-activated in order to control which coded text segments will be displayed in the respective window which lists the coded segments.
Logical Activation of Texts: It allows the user to use the values of existing variables in order to formulate logical conditions for searching and selecting, i.e. activating, only those texts for which the conditions hold.
Preprocessor: A WinMAXpro option which helps to import numerous and pre-structured (pre-coded) texts.
Text Group: A subset of the text data of a WinMAXpro project. At least one text group must be defined in order to work with WinMAXpro.

---

16 The Windows 3.1 version (WinMAX 96) is still available.

14.2 Text importing and structure

The user of WinMAXpro first creates a project to which at least one text group is assigned. One or more ASCII/ANSI text files can then be attached to a text group. WinMAXpro formats each text file and line-numbers the text automatically (the default is 80 characters per line, but the user may change this). If there are carriage returns in the ASCII file already, then these are numbered as separate lines. Line numbering is a necessary process since the display of coding as well as memo writing are attached to lines. Texts 'imported' in WinMAXpro are read-only documents and may not be edited during analysis.

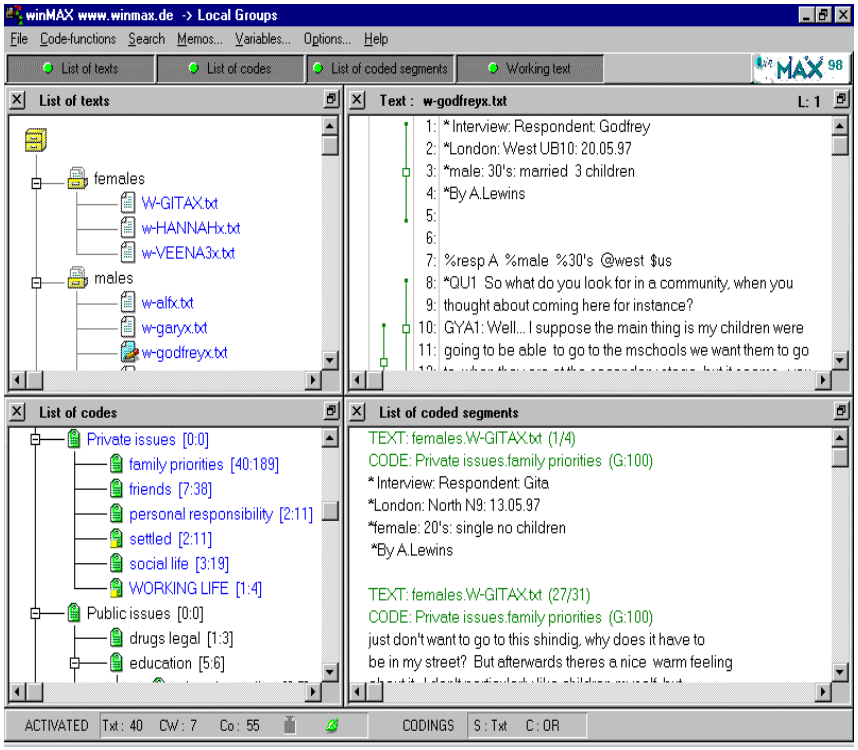
The Preprocessor functionality of WinMAXpro supports the import of already coded or structured texts, as long as the respective coded units are delimited as prescribed by the program. For example, a file containing answers to open-ended questions, each of which is preceded by the marker '#TEXT', may be imported in one go as a text group with each answer being a text of the group. Note that an ID or name may be specified for each text simply by typing it in next to the #TEXT marker. In the same way, the user may import different interviews, with each interview marked as a separate text if this is required. Furthermore, if the interview transcripts contain information about the speaker or comments to the general context of the interview, etc., this kind of information may also be imported as long as the relevant text segments are delimited within the markers '#CODE' with a name for it, e.g. speaker, comment, and '#ENDCODE'. WinMAXpro imports such structured texts in one go, stores them as separate texts assigned to a defined text group and stores and displays the provided codes in the Coding Manager window.

Note that a text imported in WinMAXpro may not be larger than 2MB.

14.3 User interface

The WinMAXpro desktop (see figure 21) comprises a menu-bar, a status bar for opening and closing the four 'standard' windows of WinMAXpro, the four windows themselves, and a feedback bar at the bottom of the desktop showing how many texts are activated, which coded text segments are shown, etc. Depending on whether all or some of the windows are open, the desktop is dynamically re-arranged to adjust to the number of opened windows.

Figure 21



One of the WinMAXpro standard windows is for listing the text groups and those texts which belong to each group (the text list window). Another window is for displaying the working text, which the user selects to open and work with. This window may show only one text at a time. A third window lists the coded text segments and yet another window is for listing and managing the codes. With the exception of the window displaying the working text, the other three each contain an object ‘manager’, called Project Manager and Text Manager, Code List Manager, and Coding Manager. Each manager is a pop-up window which includes a list of buttons for the different commands and operations related to its particular window, such as coding, activating, sorting the coded text segments, etc. The working text window contains three columns placed on the left of the displayed text: the memo column for displaying the memo icons (if memos have been created for a text

line), the coding column, for displaying existing coding, and the line number column. The user may decide to keep the columns 'open' or 'closed' during analysis.

The code column is mouse-sensitive: placing the mouse on a code icon in the respective column displays the name of the attached code. Adding more coding expands the coding column appropriately.

Coding is performed by first selecting a text segment and then clicking on a code in the Code List window, bringing the Code List Manager up and pushing the Code button. The text segment selected and coded is shown in colour.

The contents listed in the window displaying a list of the coded text segments are controlled by means of activating or de-activating texts (see section 14.1) in the Text Manager. Alternatively, the use may use the logical activation option of the Project Manager menu: This enables one to activate only those texts for whose text variables contain values which the user specifies. Chosen variables may be connected with the Boolean operators OR or AND as well as by using numerical operators, e.g. 'equal to', 'more than' or 'less than', etc., can be set for their values. Only those texts for which the specified conditions hold are then activated.

To select a string of words to be coded in WinMAXpro the user operates the left and right mouse buttons for marking the beginning and the end of a selection and for de-selecting and selecting again. The selected text segment is displayed in green. Whether this manner of text selection is more appropriate than the other depends on the amount of text to be selected, and is a matter of taste and exercise. The developer of WinMAXpro points[17] out that this manner of text selection was chosen because text segments in qualitative text analysis are often longer than the amount displayed on the screen.

WinMAXpro's on-line help is detailed and includes an alphabetically-sorted topic index.

14.4 Text coding and categorisation

WinMAXpro supports the user to create a flat or hierarchical system of codes to code her texts. Up to ten hierarchy levels may be created and the program imposes no restrictions on the amount of codes and sub-codes used. Codes may be deleted or renamed and the text segments coded according to a code can be moved to or copied under another code.

Coding is performed only on one text of a text group at a time. The user codes on-screen by selecting one text segment at a time and attaching it to a code. Coding of

---

17 Udo Kuckartz, October 1998, personal communication.

more than one segment at a time is possible only after a search for either a word or a string of words or a stem[18] has been performed using the Code All option, contained in the window displaying a list of all retrieved positions. WinMAXpro will code them all and update the information as to the number of times a code has been assigned and the total number of lines coded by it. A note of caution here: if the user hits the Code All option again by accident, WinMAXpro provides no notice that the retrieved hits have already been coded if the code they assigned is also the same, and it will code all text segments twice.

A coded text segment consists of three sets of information: (i) its related project, text group and text, (ii) its boundaries (i.e. beginning and end), and (iii) the assigned code(s). The user may assign any number of codes to a text and the number of coded segments is not restricted. Moreover, WinMAXpro supports nested coding, where parts of an already coded text segment can be assigned to different codes. The same text segment may be assigned to one code or more than one code.

WinMAXpro lets the user assign a weight to coded segments. This can be useful if one wants to code text passages which are more characteristic, in other words, carry more weight, for a particular concept one is coding for.

The user may attach memos to text lines. Codes can also be attached to memos. Moreover, each memo may contain information about its author and the date it was created. WinMAXpro provides a Memo Manager (similar to the other 'Manager' windows of the program) for memo maintenance. Memos can be filtered according to activated texts or codes, author, date, word or string occurring in a memo text.

WinMAXpro supports text variable definition, whereby the user first defines a variable and then provides a variable value for a text of a text group. Variables may be numeric or string-based, can be used for filtering the displayed coded segments or the texts activated and may be exported for further quantitative analysis.

Teamwork with other WinMAXpro users is supported by enabling the import and export of WinMAXpro projects. A helpful feature of the program is that for every text group an overview for each text contained in the group is provided, with information about when a text was last modified, how many coded segments it entails, as well as what memos are attached to it. This supports single WinMAXpro users working together.

---

[18] Note that the word stem simply means a string of characters. No lemmatisation is used in WinMAXpro.

The program also supports re-usability of coding as long as the data considered are WinMAXpro data or delimited in a way that the Preprocessor of WinMAXpro can interpret (as explained in the section on Text importing above). As reported by Kuckartz (1998) a format for data exchange has been agreed upon between the developers of three analysis packages, namely INTEXT (Klein 1997b), WinMAXpro, and AQUAD (Huber 1997). This could enable the user of WinMAXpro to import coding and coding schemes which have been developed while using INTEXT or AQUAD. As far as we know, this has already been implemented in WinMAXpro, but not in the other two packages.

14.5 Text and coding exploration

The WinMAXpro user can search for all occurrences of a word, a string of words or a word stem in a text group. Depending on whether any text variables have been activated, it is possible to use them as filters for restricting a search over a subset of the texts of a text group. Furthermore, a search may be restricted to either the texts of a text group or all coded text segments. The Boolean operators OR or AND may be used when searching for a set of words or strings of words or word stems and the user may specify whether upper- or lowercase spelling should be considered.

WinMAXpro presents the search output either

1. as one hit at a time (called Direct)
2. or in a separate window as a list with each hit displayed with the title of the text it belongs to and the line number it occurs
3. or it can save it in a file.

For the first two options, WinMAXpro offers the option to display for each hit the respective position in text. The user cannot obtain a KWIC display with all the retrieved positions shown in context. It is fairly tedious repeatedly to select one occurrence at a time and then click on the appropriate button to locate the respective text position; all this in order to see how the searched string occurs.

Moreover, if the user searches for more than one word or phrase the display of the search results does not organise the retrieved positions separately; rather, all hits are piled together in a single list. This problem may be avoided if the user saves the result of a search in a text file. The file is formatted in a way which includes each text title, the word searched for, and the context it occurs in. If it were possible also to obtain such a presentation for the on-line retrieval, this would be an improvement in the presentation of retrieval results in WinMAXpro.

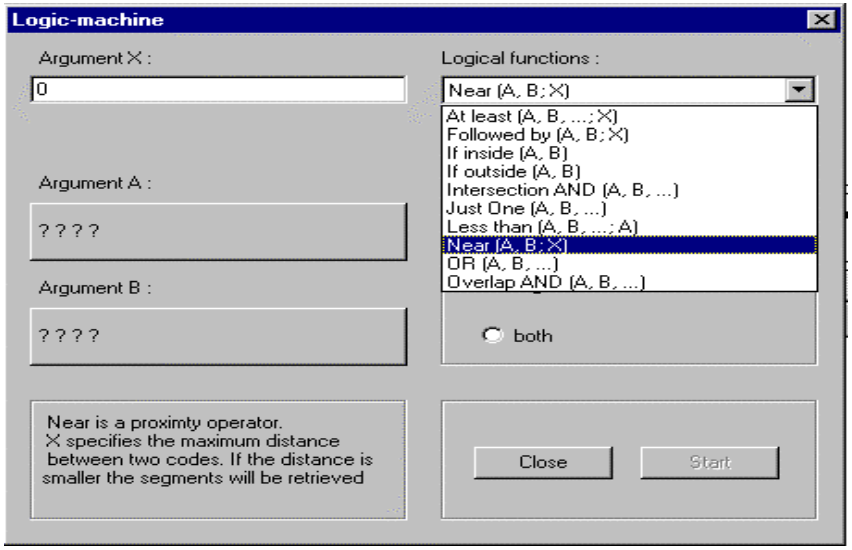
Frequency information concerning the existing coding is displayed in the Code System window with each code, followed by the number of times it has been assigned as well as the number of text lines coded with it. Moreover, a menu option provides a window listing all codes with the total of times each of the codes has been assigned to a text segment and the total of lines coded by it.

The coded text segments displayed in the respective WinMAXpro window correspond to the activated text and activated codes; whereby all texts and all codes may be activated. This way, the retrieval of coded text segments can be easily filtered by text and code. The sequence order of the segments can also be determined and the user can sort the segments either by text sequence or by code sequence.

Which coded text segments of the activated texts and codes are displayed is further controlled by means of the AND, OR, and LOGIC options of the Coding Manager. The first tells WinMAXpro to look in the corpus of the listed coded text segments for instances which have been coded according to all activated codes at a given analysis phase and display only those it has found. Selecting the OR option results in displaying all text segments which have been coded according to one of the activated codes. Regarding the final option, i.e. LOGIC, WinMAXpro looks for all those coded text segments which match a logic pattern of codes and displays them. The user defines in a window which is organised in five (logic) OR-blocks. Each of these is connected to the other with a logic AND; a code search pattern combining the two logic operators, for example (*x* OR *y*) AND *z*, where x, y, and z are codes.

More complex conditions may be formulated for searching and exploring the coded text segments by means of an additional option in the Coding Manager, namely the Logic-Machine. Here the user can specify a number of complex operations involving combinations of codes (regardless of whether they are activated). WinMAXpro offers a choice of up to 10 different ways of combining codes (see figure 22), such as *x* followed within *n* lines by *y*, or overlapping of *x* and *y*, or *x* assigned within a text segment coded by *y*, etc., where *x* and *y* code names. The number of hits found is displayed in the Logic-Machine window and WinMAXpro updates the display on the coded text segments window with the retrieved text segments.

Figure 22



14.6 Export formats and interoperability

The coding performed by using WinMAXpro can be exported for further quantitative analysis in a dBase format, which could then be imported into a statistics package such as SPSS. The user can also export the variables, their values, and the respective texts to a dBase file.

WinMAXpro will save all data related to a project not only as separate files, but also in an Archive. This system file contains all data of a specific project, that is, all texts, variables, codes, coded text segments and memos. The user may import an existing archive in WinMAXpro for further analysis.

The project in its entirety may be exported in a dBase format.

14.7 Discussion

WinMAXpro is a program which supports theory building and a large number of requirements for qualitative text analysis: text management and retrieval, coding, memo writing, hierarchical organisation of coding scheme.

We found the program easy to learn and use. Its flexible and clear-cut arrangement of different information 'units' in four different windows, i.e. the working text window, the

project and text manager, the coding scheme manager and the coded text segments manager, keeps the user close to her text and codes and assists in maintaining an overview at the coding and analysis process.

Some of the restrictions of WinMAXpro concern its text search functionality. First, WinMAXpro will not keep track of or save search strings and their results. This would be useful so that the user could re-use them or decide to code the results at a later stage in analysis. Secondly, if a number of words (phrases) as opposed to one word (phrase) is searched for, the resulting list is not displayed sorted according to the different words; rather, a single list with all hits is provided. Thirdly, after performing a text search, one must select separately each single position where a word has been found in order to see it in context. This makes viewing the retrieved instances wearisome. Fourthly, the user cannot obtain an on-line view of all the respective positions displayed in context after performing a search. Instead, it is possible to save the retrieved instances in a file as a KWIC list. This file may then be directly imported to the program. In terms of text exploration, the possibility to obtain such a view on-line and inter-linked to the text display, so that a selection of a text segment also causes the respective text and its exact text position to be displayed, would be an improvement.

The search functionality for displaying the coded text segments, with the logic operators and the variety of code combinations which may be specified in the Logic-Machine, is an important feature of WinMAXpro and a substantial help for exploring the existing coding.

The logical activation feature helps to control and filter appropriately which coded text segments should be displayed. Future development could include the possibility to create a new text group, a sort of a sub-corpus of texts, consisting of only those texts resulting from the logical activation operation.

14.8 References and further information

The demonstration version we have used for this review comes with a tutorial file. A detailed manual is available for the full user version (Kuckartz 1998), available in both a English and a German version. For a review of earlier as well as current versions of WinMAX, see Weitzman/Miles (1995), Kelle (1995), and Kelle (1996). Tips for working with WinMAX are given in Kuckartz (1998).

Information on WinMAXpro, including a trial version of it and a detailed bibliography, is available from the Web site: http://www.winmax.de. The developer of the program organises workshops for WinMAXpro users and a user conference takes place every year.

15. WordStat

WordStat is a module of the statistics program SIMSTAT from Provalis Research. It is for the automatic as well as interactive content analysis of mainly survey texts, i.e. answers to open-ended questions, but, also, of other textual data such as short advertisements.[19] WordStat runs under Windows 3.x, NT, and 95.

WordStat's approach to content analysis is dictionary-based: automatic coding is based on already existing or new user-defined dictionaries. The user can, furthermore, code interactively by means of assigning unique keywords (codes) anywhere in a text and retrieve this coding - if the keywords are included in the dictionary. WordStat in combination with SIMSTAT supports the user in assessing the reliability of coding by providing eight different inter-raters agreement measures.

WordStat offers general statistical text information, such as word counts for all words or only those words which are included in the dictionary used, category counts or calculation of word by word co-occurrence matrices, and provides KWIC as well as Code-In-Context displays.

The coding results can be used as new variables representing the occurrence of words or categories in statistical analysis (such as cluster analysis on words or cases, principal coordinate analysis, correspondence analysis, multiple regression, etc.) in SIMSTAT. The data can be imported from and exported to different file formats.

For the review, we used the 30-days test copy of SIMSTAT (version 1.1) and WordStat (version 1.1b).

15.1 Terminology

Exclusion Dictionary: A list of words which are not considered for the analysis; the so-called stop word list.
Inclusion Dictionary: A list of all the categories used for coding (note that the WordStat Help text uses the word "words" instead of the word categories).
Memo (or Memo Field) and Record: A text unit which accepts up to 32K of text. WordStat can analyse text stored in memo field (up to 32K) or alphanumeric fields (only 255 characters). A single record is usually associated with a respondent and can contain answers to different open-ended questions.

---

19 The module was implemented by Normand Peladeau.

Substitution Dictionary: The dictionary categories and their entries. The categories included may not only group words denoting a 'semantic' category, but they may also be groups of word forms having the same lemma.

15.2 Text importing and structure

In order to use the WordStat content analysis module, the text to be analysed must be imported in SIMSTAT. Machine-readable texts may be imported in SIMSTAT as spreadsheet files produced by Lotus, Excel, or QuatroPro, but the maximum text length for each cell is 256 characters. Spreadsheet files containing longer texts, should be first saved as tab-delimited text files and then be imported in SIMSTAT. In addition, short texts, containing no more than 256 characters, contained in dBase or Paradox database files, may be imported in SIMSTAT.

Text files with longer texts and several numeric and alphanumeric variables may be imported as comma, or tab delimited files. The first line of the file contains the variable names separated by commas or tabs, and the remaining lines contain the numeric or alphanumeric values for each variable, separated by commas or tabs.

The user can choose several 'memos' or alphanumeric fields. Each field contains texts such as answers to different open-ended questions and either may be analysed as if they were a single text unit or analysed separately. This means that if the texts are to be analysed together, they become a single text unit. A memo field can contain up to 32K of text.

A memo has a number of values for the variables used, for example answers to open-ended questions texts may have a project ID, a text/respondent ID, and a question ID. Other text structure within a memo field is not possible. Numerical variables which function as independent variables may be used as filters for grouping texts together.

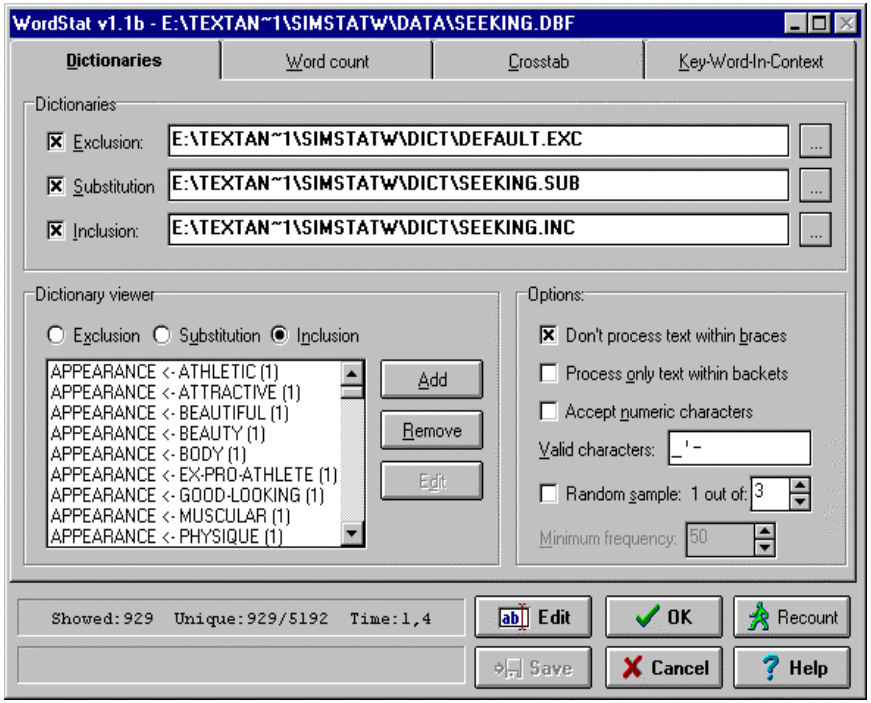
WordStat does not differentiate between upper- and lower-case and accepts files in either upper- or lower-case; a feature which is, of course, problematic for languages such as German.

Interview transcripts or answers to open-ended questions which are not already in machine-readable form may be directly typed in, in SIMSTAT. Text which has already been imported in WordStat may be edited and a spelling check option is provided.

15.3 User interface

The WordStat working environment is organised according to four virtual 'pages' or 'folders' shown in figure 23.

Figure 23



Each WordStat page enables different analysis and coding operations. The operations are organised either in pull-down menus or as type-in fields. The first page, called Dictionaries, enables the setting, display and maintenance of the Inclusion, Exclusion, and Substitution lists. Other options concern, for example, the handling of special characters or numbers in the texts or how the program should handle text which is delimited by brackets.

On the second page, called Word count, the user can set the options for the display of the word frequency index: either display all words or only those words which are contained in the Inclusion dictionary or those words which are not contained in the Inclusion dictionary. Words can be sorted according to alphabet or frequency of occurrence, and additional statistical information may be chosen to be displayed, e.g. similarity index or agglomeration order. One way of graphically displaying a cluster analysis of all included words

(called categories) is the Dendrogram option. Here a tree is displayed based on the proximity of these words (similarity index or agglomeration order). This display technique is fairly poor in text mode.

The Crosstabs page manages the cross-tabulation process in WordStat. Cross-tabulation can be calculated between record occurrence or word frequency and other words or socio-demographic variables (if available). The results may be presented as sums or percentages and may be sorted in different order. Upon selection of one of the sums or percentages for a category, the user may use the pop-up menu with options for

- renaming the category
- removing it from the Inclusion dictionary
- moving it to the Exclusion dictionary.

This way, the dictionary may also be maintained from the Crosstabs window.

On the fourth page, called Key-Word-In-Context, the user can specify which dictionary should be considered for the KWIC display: if the Inclusion dictionary is specified, then a list of all its categories is provided in a combo box. The user can select one of the categories and obtain a 'Code-In-Context' display of all occurrences of the words belonging to the specified category. Alternatively, the user may specify that Included words should not be considered. The user may then choose from the menu of the combo-box with all 'not-Included' words a not-Included word and obtain a KWIC display. In a third option for obtaining a KWIC display, namely user-defined, the user types in a keyword for the KWIC display. If an Exclusion dictionary is used, however, this keyword should not be one of the words contained in it. The available sorting criteria for the KWIC display are: sort by record number or keyword-and-before or keyword-and-after or keyword and socio-demographic variables, or demographic variables and keywords. The KWIC display is a table containing a number of columns for the record ID, the text before the keyword, the keyword itself, the text following the keyword, and the other available IDs or values to the specified variables. The borders of the table may be expanded or narrowed.

At any stage of the analysis and on each WordStat page both the text data and the three dictionaries can be edited. Only one text, that is, one 'memo' field may be displayed and edited at a time.

The Save option of each WordStat page saves always the same output; no matter what the specific purpose of each page and output of operation is. One would expect that the different pages save different kinds of information.

On-line help is available in WordStat and a 'quick tour' for new users is also provided.

15.4 Text coding and categorisation

The coding process uses three sources of information: the Exclusion dictionary which, in effect, consists of a stop word list; the Substitution dictionary which holds the categorisation, i.e. which categories there are and to which category a word belongs; and finally, an Inclusion dictionary which contains a list of all the categories to be considered for the analysis. Note that the existence or usage of these dictionaries is optional: if the user chooses to use no dictionaries at all, the analysis process is reduced to generating just a frequency of word analysis and no coding is performed.

WordStat codes the imported text automatically, based on the provided Substitution dictionary. Furthermore, the user may code interactively by assigning unique keywords anywhere in a text. The retrieval of manual coding is possible only if the keywords are included in the dictionary.

Coding in WordStat is single word-based and the inclusion of phrases in the dictionary is not possible.[20] The WordStat module comes with an already constructed Exclusion dictionary for English which can be modified. The other two dictionaries are constructed by the user. WordStat does not support multiple coding.

The cross-tabulation result displayed in the Crosstabs window may be used to maintain the dictionary: the displayed categories may be renamed, sent to the Exclusion list or removed from the Inclusion dictionary.

SimStat supports the user in assessing the reliability of coding by providing eight different inter-raters agreement measures. The assessment is based on the presence or absence of a specific code and is calculated for each code.

15.5 Text and coding exploration

WordStat offers three different groups of analysis functions: word/category count, cross-tabulations, and KWIC lists.

Frequency of occurrence information provided for each word in a word list can be filtered so that only 'categories' are considered. This means that either all words or only those words which are contained in the Inclusion dictionary or those words which are not contained in the Inclusion dictionary are displayed. If a Substitution dictionary is specified and chosen as a filter, then only those 'words' which are the lemmata or the categories are included; for example, if the Substitution dictionary option – or better filter - is selected,

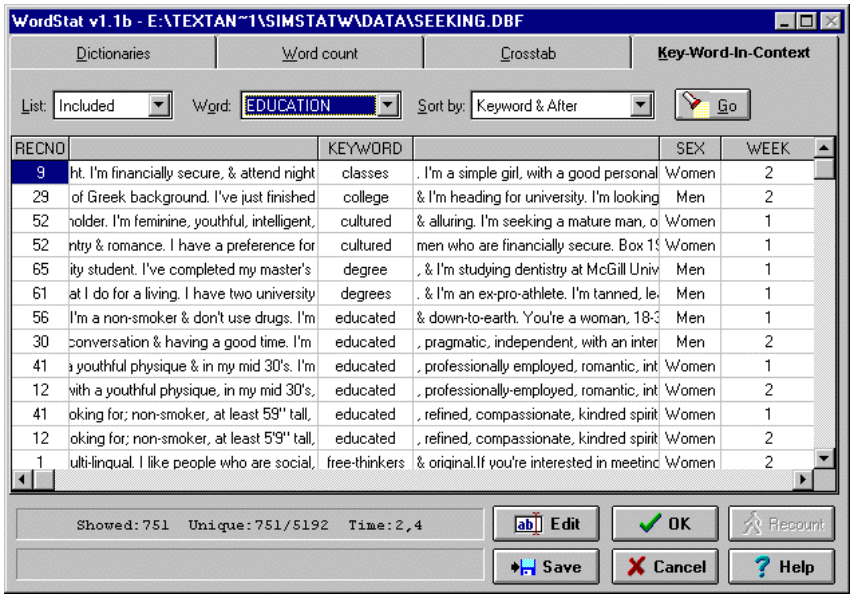
---

20 Note that the version released in October 1998 supports also the coding of phrases if these are delimited appropriately, e.g. by means of the hyphen or underscore character.

the resulting word list does not contain the words ‘art’, ‘artistic’ and ‘arts’ if all three have been ‘substituted’ by the category (or lemma) ‘art’, but rather only the word ‘art’ is shown. If no Substitution list is specified each word type is counted and contained in the list. The word lists can be sorted alphabetically, by higher or lower frequency, by word endings or by record occurrence.

Beside word counts, WordStat offers cross-tabulations of word frequencies or record occurrences with other words or socio-demographic variables. The results can be presented as sum values, row or column percentages, or total percentages. They may be sorted alphabetically, by frequency of occurrence or by record occurrences. WordStat facilitates the assessment of the relationship between a specified independent variable, e.g. the sex of the person responding to a question, with the utilisation of each category of the Included dictionary. This is achieved by calculating twelve different association measures (the user may choose one of the options in the Statistic drop-down list), for example chi-square, likelihood ratio or Pearson’s R statistics.

Figure 24



WordStat searches for all occurrences of a category; for example, searching for the category 'education' will provide the KWIC list shown in figure 24 with the words assigned to this category, including for example, 'intelligent', 'learning', etc.

If the option User Defined is selected, then the user may type in the Word field a word (or category) to be searched for. This switching between category and word is fairly confusing. Wild cards can also be used for the search.

A limitation in the searching functionality is that it is not possible to search for a string of words, e.g. 'go* out'. Moreover, in the user-defined search modus, the user cannot search for words included in the Exclusion dictionary (e.g. 'for', 'to') if the option that an exclusion field should be used for the analysis has been selected. The KWIC display which results after searching in the user-defined mode does not provide any information about whether, and if so how, the displayed word has been categorised. In order to see this the user must open the text field where the word is highlighted in colour. This operation has to be performed for each text whose text segment(s) is included in the KWIC display. Moreover, no other operation is possible while the window displaying a specific text is open.

15.6 Export formats and interoperability

As WordStat is fully integrated with the statistics software SIMSTAT, it allows the user to export data to dBase, Paradox, Excel, Lotus 123, Quatro Pro, SPSS/PC+ or as comma or tab-delimited files. When saving data as ASCII files a maximum of 500 variables will be exported, whereas when saving in a spreadsheet format a maximum of 255 variables may be exported. Filtering options are available for exporting only a subset of cases.

The WordStat output files as dBase files can be used, for example, as input in SPSS. The codes are stored as category frequencies or as dummies (0/1=category mentioned/not mentioned) and are assigned to complete text units.

15.7 Discussion

WordStat is a quantitative text analysis program for coding and analysing answers to open-ended questions or similar short texts. WordStat may be appropriate for medium survey analysis projects involving textual data with simple, unsophisticated text structure, and where simple and unsophisticated categorisation is required. As it is fully integrated with the statistics software SIMSTAT it has the advantage of facilitating the seamless further statistical processing of the coded data. This allows the users to perform other (advanced) analyses, such as cluster analysis, of the numerical results obtained from content analysis within the same application. The fact that as a SIMSTAT module WordStat may be initiated only when using SIMSTAT may be a disadvantage for users of other statistics software. However, the developer of WordStat is considering preparation of a

standalone version (personal communication with Normand Peladeau) which could be interesting for a large number of users.

Regarding the WordStat working environment, its four-page organisation, with each of the pages having a separate functionality, may help the user to obtain a quick understanding of how the module works as well as learn to use it fairly fast.

The connection between coded words and text is rather cumbersome in WordStat: for example, only one item of the KWIC display can be selected at a time in order to obtain a display of the whole text in which this occurs. A new window displays the particular text and the user may select only one of the categories. The occurrences of this category are then displayed highlighted in colour. This window must be closed in order for the user to be able to return to working WordStat.

The fact that the term 'word' is used indiscriminately is confusing in WordStat. Sometimes it means a word type, sometimes a category and sometimes a lemma, depending on the dictionaries used.

An important disadvantage with regards to coding in WordStat concerns the fact that analysis and subsequently coding is single word-based: the building of categories is word-based and phrases are not allowed. One way round this which, however, is not a serious solution to the problem, is to edit the original text and use hyphens or another valid character to join these words together.

In October 1998 a new version (1.2) of WordStat was released. The new version includes such new features as the possibility to have an integrated thesaurus for English which can assist the creation of categorisation schemes, an improved display of dendrogram, and flexible record filtering on any numeric or alphanumeric field and on code occurrence with AND, OR, and NOT Boolean operators.

15.8 References and further information

Information about WordStat (and SimStat) can be obtained from http://www.Simstat.com/wordstat.htm. One can order the program or download a 30-days test copy from the same Web site.

The 30-days test version contains a guide through the system and on-line help.

PART 3: REVIEWING THE REVIEWS

We set out first to provide a detailed and up-to-date account of the spectrum of available text analysis software and, second, catalogue the kinds of support the selected software offer to the user. The purpose of reviewing the software is to record the tendencies both in functionality and technology, and identify the areas where more development is needed. So far we have presented and discussed each selected software according to its functionality for importing and exporting texts and coding, recognising particular structural elements of a text, for creating categorisation schemes and coding texts, for exploring the text material as well as the existing coding and its user interface. Based on the provided accounts for each program, the third part of this book reflects on the 'state of the art' in text analysis software, summarises differences and commonalities between the presented programs, and discusses some of the lacking features and aspects related to future development.

The user of text analysis software today has at her disposal a considerable variety of software. Each of these supports text analysis to a greater or lesser extent by providing means for organising and managing texts in projects or groups according to particular features, looking for how particular words or phrases or coded text segments occur in a single text or in a collection of texts, creating and maintaining coding categories, keeping notes and memos on text or categories or coded text segments, obtaining different 'views' of text and existing coding, and exporting the coding in different formats either in order to use it as input to other software, e.g. statistics, or to present the results of the analysis, e.g. reports. One might assume that someone engaging in text analysis today is spoilt for choice; and, indeed, depending on the task at hand, it may be so. Still, there are text analysis tasks and methodological approaches which, we believe, are not supported enough or they lack functionality which has yet to be developed or integrated into text analysis systems.

In general, the reviews so far show that a single text analysis software package alone

- either supports only *some* of the operations required for a particular text analysis project. For example, the strengths of HyperRESEARCH are coding and hypothesis testing, rather than the general text search-and-retrieval mechanisms, which are not supported
- or it may not support certain operations efficiently or elegantly or user-friendly enough; for example, the limited support of CoAn and TEXTPACK for the exploration of coding.

The above points would not be necessarily problematic if it were possible to use two or even more software packages for a single text analysis project in a seamless and user-friendly way. In fact, for certain projects it can make sense to work in an 'integrating' manner, that is, to apply different types of approaches and techniques for, e.g. testing purposes or for juxtaposing and comparing the analysis results obtained. Besides, there are projects which employ a mixed model methodology (for mixed methodology see e.g. Tashakkori/Teddlie 1998) for which this would be advantageous.

Unfortunately, the text analyst in her role as a researcher, student, teacher, lecturer, professional, evaluation expert, etc. or as psychologist, linguist, social scientist, scholar, journalist, etc., wishing to integrate different modes of coding or different methods is not spoilt for choice yet. The merits of different types of methodology and approaches are

- either not combined in a single software,
- or cannot be coupled by using two or more text analysis programs for the same body of texts, re-using, and perhaps augmenting existing coding or extending categorisation schemes or experimenting with different text exploration techniques
- or, finally, can be linked, but only if the user has invested or is willing to invest a considerable amount of work in pre- and post-processing.

There is a clear lack of support for standards which could contribute to supporting or improving the interoperability between text analysis programs. This lack stretches out to missing or weak interoperability of text analysis programs with other software which can enhance, supplement or extend the analysis process, e.g. statistics, language parsers, lexical databases, semantic network builders, etc.

Of course, programs differ according to the type of text analysis they have been designed to support, i.e. general text analysis, quantitative or qualitative analysis. More interestingly, though, there are fundamental differences even within the same type of analysis approach supported. The amount and degree of variance increases if we also take into consideration text analysis programs which have not been reviewed here because they are all MS-DOS-based, e.g. Ethnograph (http://www.qualisResearch.com/), INTEXT (Klein 1997b), Kwalitan (http://www.kun.nl/methoden/kwalitan/), MECA (Carley/Palmquist 1992), TACT (http://www.epas.utoronto.ca:8080/cch/TACT/tact0.html), PLCA (Roberts 1989), etc., which we have, nevertheless, examined and which also contributed to shaping the group of 'primary' operations for text analysis discussed in the first part of the book. As a means to illustrate the above-mentioned variance let us take the coding unit in qualitative-oriented programs: this may be a line or a word or a sequence of words or the whole text or document. Consider also the dictionary or dictionaries used for automatic coding in quantitative programs: they can be user-defined or system internal or both. We

believe that the choice of one or more programs for a specific text analysis task or project is far better served when not solely the general type of text analysis approach supported is considered, but rather when the user takes into account the *ways* by which analysis operations are supported by the program or programs and how these affect the results of analysis.

The discussion of both differences and commonalities between the programs, and the lacking features and aspects related to future development is organised according to the following topics: text import mechanisms and the kinds of text structure software can handle, the ways software support categorisation and coding, data export mechanisms, kinds of linguistic processing and information used by software to enhance analysis, kinds of text exploration support with search for and retrieval mechanisms for texts, text strings, codes and memos, the user interface and aspects concerning whether and how different modes of analysis are integrated as well as aspects related to the generality of use of software.

1. Importing text data (text sources)

All fifteen text analysis software reviewed expect ASCII/ANSI text files as input. The way text is entered into the program for analysis differs: ASCII text files may be stored externally to the program, e.g. HyperRESEARCH, or they must be imported and translated into some internal representation which the program uses, e.g. TATOE. In the latter case, if the text data have for some reason been modified then the text needs to be re-imported.

Programs differ according to either whether they expect that the text data to be analysed are read-only and not to be edited any further, e.g. AQUAD, TATOE, WinMAXpro, or whether they support the editing of the original data during analysis, e.g. CoAn, KEDS, NUD*IST. ATLAS.ti stands somewhere in the middle: it expects that the documents to be analysed are read-only, but at the same time it allows the user logged as 'administrator' to switch to edit mode. Adding or removing a text which has already been coded can result in corrupting the hyper-linking between coded positions. This means that if the user in a later stage activates a coded text segment she runs the risk that ATLAS.ti may highlight wrong passages in the text. On the one hand, allowing the combination of editing and coding can be advantageous if changes should be made to the text data used; especially if this does not invalidate subsequent coding and the system updates the existing files afterwards as supported by NUD*IST. On the other hand, there are indeed advantages in using read-only documents for analysis: the possibility of multiple coding, the sharing of the same documents among users, the direct processing of legacy text data or CD-ROM texts which are typically read-only documents, etc.

The packages reviewed differ with respect to the kind of text structure they can recognise and work with as well as the ways of delimiting structural units. It must be clear from the

review that not all programs delimit the same structural elements. To give an example: WinMAXpro structures texts according to line numbers, whereby numbering is performed automatically by text import in WinMAXpro; if hard carriage returns are included in the ASCII file, these will be interpreted as new lines. On the other hand, NUD*IST works with Text Units as opposed to WinMAXpro and AQUAD working with lines. NUD*IST Text Units are delimited with hard carriage returns, usually used for delimiting paragraphs. If a Text Unit is other than a paragraph, e.g. a sentence, a line or even a word, the text must be edited so that carriage returns are inserted after the end of each unit.

The text files imported into NUD*IST may contain a header and a sub-header: the former involves information about the project, the speakers of an interview, etc., and the later serves the purpose of dividing the documents into sections (each section is delimited by the '*' character). The user must determine the text structure prior to importing the texts and, hence, pre-processing may be required to structure the files appropriately. In contrast, no header or sub-header information is recognised by HyperRESEARCH. In fact, HyperRESEARCH can identify no other structural elements of a text than the text itself.

CoAn and TEXTPACK input ASCII files, which may contain identifiers (IDs) denoting text units. Both programs can work with three 'levels' for IDs. Such levels can be used to indicate a textual element or feature, e.g. a paragraph or the questionnaire number of a text which is an answer to an open-ended question, or meta-textual information, e.g. speaker, text header. The ID values denote the features which are coded. IDs are delimited by special characters, e.g. $ or %. Of course, the text must be pre-edited accordingly before analysing it with the respective programs.

Two points concerning text import are worth discussing here: first, the specific mark up for delimiting structural elements of texts and second, which structural elements can be represented (and processed) by which program. The majority of the reviewed programs neither make use of a standard for the encoding of the different types of texts whose analysis they support nor conform to or exploit the SGML/XML standard for document representation. TATOE supports the import of XML-encoded data, but in a restricted way using XML as a kind of system internal exchange format for text data. Kuckartz (1998) reports on the agreement to develop an interchange format between three text analysis packages, namely INTEXT, WinMAXpro, and AQUAD for qualitative data analysis. Texts, codes, and coded text segments may be represented whereas other information types, such as memos or variables cannot. The information is stored in a dBase file. To date this functionality has been implemented only by WinMAXpro.

If a package recognises specific structural elements of a text, i.e. text unit, header, comment, it usually prescribes idiosyncratic encoding means for marking these, e.g. special

characters used as delimiters, carriage returns, etc. The consequence of this are two-fold. On the one hand, although considerable amount of time and effort is invested in the preparation (pre-formatting, decisions about which structural elements are important for analysis, etc.) of a text or body of texts for analysis, this text material can hardly be re-used with another program or for another analysis project or secondary analysis purposes. The user is hindered in using other programs or must engage once more in the same task; a time-consuming and possibly frustrating exercise

On the other hand, if no standard text encoding is used, the possibility for losing information increases, since it may not be possible to represent particular elements of text structure (for instance, comments which should be excluded from or included in text analysis proper, existing coding, etc.). Having said that, it must also be stated that even if a standard is used one still runs the risk of losing information, since the quality and flexibility of the standard plays a crucial role.

The review of the selected programs has shown that often when a finer text structure is required this is may not supported by a program. To illustrate this, consider the DIMAP-MCCA content analysis module. Say that the text data to be analysed is a corpus of three presidential debates, each of which involves four speakers. The program will not support the definition of a structure unit for the date or number of each debate, as well as for each one for the speakers (three candidates and a journalist monitoring the discussion) in a single document, since only one of the two may be explicitly specified in the document. One way round this is to create separate documents – one for each debate – but if we want to use the analysis apparatus of DIMAP-MCCA for one document containing all the debates together with the speaker identification and structuring, this will not be supported.

Let us use another example in order to illustrate our point, but without taking into account the functionality of a particular program. Say that we want to analyse an interview text and the analysis to be performed concentrates at an initial phase on the examination of the questions or comments of the interviewer and the answers of the interviewee(s), and at a later phase the examination of text with questions only or only the answer texts. If the questions and answers are not represented as separate elements in the text file to be analysed, then the program will not know how to switch to either one or both of them in order to code or search for a text segment within one of the structural elements.

In other words, the types and the degree of detail of text structure supported by a text analysis software, that is, what structural elements of text are represented, recognised and operated on, is a deciding factor for the types of operations which can be performed: only explicitly represented elements can be searched or operated on.

Some of the reviewed programs impose restrictions on the size of a text unit or the automatic structuring in text units or text lines. For instance, because HyperRESEARCH cannot access more than 16,000 characters at a time (about 3,000 words), the user must separate long text files into 'pages' by placing a tilde on a separate line to mark a HyperRESEARCH page break before starting to code. DICTION does not work with long texts either. The program splits texts which are longer than 500 words in text passages of 500 words. The reason behind this probably concerns the calculation of standardised scores of DICTION. The statistical basis for this restriction is not clear and we cannot see a 'semantic' basis for this.

It has already been mentioned that line numbers are used by both AQUAD and WinMAXpro for coding and displaying texts. Whereas AQUAD expects that a maximum of 60 characters occur in a line, the user of WinMAXpro may customise the line length. The user of AQUAD may code line by line in contrast to the user of WinMAXpro, who can select a text segment whose start or end is in the middle of a line. The coding in WinMAXpro is referred to the respective category by the number of lines it has been assigned to.

There are two kinds of supplementary data which programs may import or take into account for analysis: socio-demographic variables and categorisation schemes. The former concern information specific to age, sex, profession of either the speakers of an interview or of the persons answering an open-ended question of a questionnaire. NUD*IST, TEXTPACK, TextSmart, WinMAXpro, and WordStat support the integration of such information, typically as numeric variables, during analysis. Support for the import of categorisation schemes is advantageous if the user wants to work with an already developed categorisation scheme or a given list of codes. An ASCII file containing a list of categories can be imported in ATLAS.ti, Code-A-Text, and TATOE.

So far we have only discussed the import of source text data which contain either no coding or some coding for delimiting a number of structural elements of the source texts. Source text data which are already coded, possibly by using some other software may, however, also be input to a text analysis program. To a certain extent TATOE and WinMAXpro support the import of specific kinds of coded text. Regarding TATOE texts which have already been imported may be saved, analysed with the Morphy program (Lezius et al. 1998) for part of speech information and be re-imported in the same textbase. In WinMAXpro the coded data must be in a certain format: the start and end of a code must be delimited with a special character and the text must be included between the start and end delimiters. Note, however, that this procedure means that a code is assigned to the whole text and not to a specific part of a text, say a single line. By means of this procedure, one can import into WinMAXpro answers to open-ended questions in a single step.

2. Creating categories and coding text

What is meant by a code or category varies for the software reviewed. Although generally speaking all categories are in a sense interpretative, as far as the reviewed programs are concerned, some codes are used to semantically categorise the occurrences of a phenomenon according to the interpretative framework the analyst is working with or attempting to construct, whereas other codes are used to attach information external to the semantic interpretation, such as who is talking in a discussion text, when the discussion took place, etc. In the latter case, codes denote text or case variables, e.g. the age of the interviewee or the date an interview was taken or the name of the newspaper whose articles are analysed. Categories in NUD*IST are called nodes and are organised in an Index Tree, which is a hierarchy of the categories created in NUD*IST for coding. Nodes may embody both kinds of codes, i.e. semantic categories and variables. There are no overarching categories or schemes to keep these separate. A possible disadvantage of such a combination is that the NUD*IST Index Tree may grow tremendously which will make the handling and keeping of an overview of the interpretation framework difficult.

A different type of category for coding is what Code-A-Text calls numeric scales which are based on frequency of occurrence information. Numeric scales are automatically created by the program.

The identity stamp of a category differs depending not only on the particular program, but also on the variant of text analysis the program aims to support. In particular, categories in qualitative programs and some quantitative programs, such as DICTION, TextSmart, WordStat, are only defined by a name. These are in contrast to the categories of some quantitative-oriented programs which are also given a numerical value, typical of content analysis tradition. The output of analysis of such programs is information as to whether or not numerical values have been used for coding and, if so, how often. Such information can be further statistically processed. A NUD*IST node has, alternatively, a name and an 'address', i.e. the reference of the node in the coding hierarchy as a numeric value.

Categories may be elements of a set, i.e. a categorisation scheme or a dictionary, or they may be organised in hierarchies or networks. The general tendency is that programs which support the building of a categorisation scheme and theory building also support the construction of category hierarchies or networks of categories next to the creation of categories which do not belong to a scheme, e.g. ATLAS.ti, NUD*IST, TATOE, WinMAXpro. Only one package reviewed, namely, ATLAS.ti provides functionality for the definition of other than hierarchical relations to hold between two codes. The semantic relations offered in ATLAS.ti are: cause_of, part_of, is_associated_with, contradicts, is_a

and is_property_of. If more relations are required, the program guides the user through a menu of options for defining new relations.

Categories in ATLAS.ti may be organised in Code Families, whereby a single category may belong to more Families. This way the user organises codes and their respective coded segments in meaningful groups and at the same time she can combine codes from different Families; a helpful feature both for operations concerning text exploration and further coding. In comparison, categories in TATOE are organised in schemes and each category belongs to only one Scheme. There are no relations holding between Schemes. The definition and usage of multiple schemes may be seen as a way of creating different layers of interpretation on the same text corpus. The user of TATOE may define search patterns, i.e. rules for searching for sequences or alterations of different categories of different schemes also including strings, and then, if required, assign all the retrieved instances to a category of a given Scheme. This way TATOE supports 'multi-level analysis' in the sense that different types of coding, e.g. interpretative coding and part of speech categorisation, provide additional means for searching the text data, validating coding, and working towards theory testing and building.

Dictionaries used for the automatic coding of texts typically contain groups of words each of which is assigned to a category. A dictionary has a number of categories with no relation to each other, i.e. they are just elements of a set as, for instance, the categories of the dictionary of DICTION or they may be organised in super-categories.

Contextual parameters or overarching categories may be used in combination with specific dictionaries for guiding the analysis and scoring process. The DIMAP-MCCA dictionary, for instance, uses four contextual parameters, namely, Traditional, Practical, Emotional, and Analytic to calculate context scores (called C-scores*)*. The idea here is that C-scores capture the structuring of ideas and may determine different social contexts. DICTION, on the other hand, makes use of five super-categories, i.e. Activity, Certainty, Realism, Optimism, Commonality as 'the main' dimensions or features for describing the language of political speeches (Hart 1985).

Dictionaries may also differ with regards to the types of entries they contain. Consider, for example, the dictionaries and the respective rules used by the KEDS text analysis program for the automatic as well as semi-automatic analysis of event data as these are found in English news lead sentences. KEDS uses three types of information for verbs, actors (mainly nouns and proper nouns referring to political actors) and phrases for distinguishing between the different meanings of a verb and for providing the rules for locating the source(s) and target(s) of an event within a sentence. This information is stored in three separate files, one for each type. KEDS focuses not only on the identification of types of

concepts in a text, but rather on identifying and explicitly encoding the connections between the concepts of a news lead sentence.

Text analysis programs differ according to the text unit a code is assigned to. For example, the coding in TextSmart assigns each text to one or more categories. Although the coding process is based on a word list, the result of the coding is the categorisation not of the respective word in a text, but rather of each text according to the specified set of categories. Similarly, DICTION does not code a word or a string of words of a text, but rather, categorises a text according to a number of variables defined by the developer of the program specifically for the analysis of political discourse. WinMAXpro and AQUAD assign a code to one or more text lines. The assumption which appears to be made concerning line-based coding is that text lines can be significant or carry some semantic weight with regards to coding. As far as NUD*IST is concerned, a code is assigned to the Text Units as these have been specified by the user (see also section 1 of this part). Whereas the smallest unit to be coded in TATOE is a word, ATLAS.ti and HyperRESEARCH support the coding of a string of characters which can include only part of a word. The coding unit of TATOE may not be longer than a TATOE 'paragraph', which can correspond to a text paragraph or the title of an article, a short answer to an open-ended question or a dialogue turn. CoAn, TEXTPACK, and WordStat code words or word sequences as these are specified in the dictionaries they use for automatic coding. KEDS codes words or words groups it identifies according to the dictionaries of proper nouns as well as of verbs and verb phrases it uses as types of sources, targets, and events. Free selections of text segments for coding are supported by ATLAS.ti or HyperRESEARCH.

Programs which code texts automatically on the basis of a dictionary differ with regards to whether the dictionaries are hard-coded, integrated in the system and have been prepared by the developer(s) of the program, e.g. DICTION and MCCA, or whether the dictionaries reside outside the system. In the latter case, the user creates and provides the dictionaries used, e.g. Code-A-Text, CoAn, TEXTPACK. Beside this distinction, programs may offer functionality for the user to modify the supplied dictionaries (for example KEDS or WordStat) or only allow the addition of user dictionaries to the ones mainly used by the system, as DICTION does.

For reasons of processing and storage efficiency or for more targeted analyses, programs use stop word lists. These are lists of words whose content is not assumed to be important for the analysis purposes and, therefore, should be ignored during analysis. A stop list consists typically of frequently occurring function words , e.g. the words *and*, *the*, *to* for English. The Excluded Terms list of TextSmart or the Exclusion List of WordStat are examples of stop word lists. Stop word lists may either come with the program and may be modified on- or

off-line by the user, or the user may specify a file of stop words, as is the case with Code-A-Text or TEXTPACK. Note, however, that the role of a stop word list is not the same for all programs. Rather than for coding purposes such a list may be used for performing more targeted text exploration operations instead: the WordCrunch function operation of ATLAS.ti generates a word list containing type/token ratio information and the frequency of occurrence for each word of a Primary Document, i.e. a text assigned to a project. The user may refer the program to an external file storing a list of stop words and ATLAS.ti will not consider these words when counting word frequencies and will not include them in the list it generates of all words occurring in the specified text.

The development of stop words lists which are suitable for the aims of a specific analysis project is not a trivial matter. A list created for a particular research purpose may not be appropriate for another analysis purpose since the high frequency of occurrence of a number of words alone cannot determine whether each word should be included in the stop list and, therefore, not be considered for coding purposes. For example, for one text analysis project the frequently occurring words *I*, *my* or *me* may not be relevant and, thus, be included in the stop word list. However, for another text analysis project examining 'self reference' these words are significant for the analysis and should not be included in the stop list. Software should aid the user in creating such lists (if these are required for analysis) by providing, for instance, word frequency lists for a specific text, word distribution, and co-occurrence information, etc.

For the majority of the software supporting automatic coding according to a dictionary, the merging of dictionaries is an operation which is not supported by the text analysis software itself, e.g. KEDS, WordStat. A promising way to avoid this shortcoming is put forward by the DIMAP software with its content analysis module: DIMAP is a dictionary development software and has the content analysis program MCCA as an integrated content analysis component of the software. Even though the strengths of this approach are still to be demonstrated, in principle this may prove profitable for dictionary-based content analysis.

This review does not include a program for either the type of linguistic analysis described by Roberts (1997b) or Franzosi (1997) or for map analysis described by Carley (1993) since no Windows versions of programs supporting these types of analysis, i.e. PLCA, PC-ACE, and MECA respectively are available yet. For linguistic/semantic analysis the identification of key concepts or categories, such as actor or subject of an event, is not enough, but rather the linguistic/semantic connections between the concepts of a sentence or clause must be explicitly encoded on the basis of a semantic grammar. KEDS is the only program we have reviewed which also aims at the extraction of a set of related con-

cepts (i.e. targets and sources of an event) according to a set of rules. Given a set of texts and a set of concepts map analysis aims to determine for each text whether and which of these concepts occur, as well as determine the cognitive relations between the occurring concepts. From the programs reviewed only ATLAS.ti allows the user to specify manually different relations to hold between codes and display them as a network. Nevertheless ATLAS.ti, is probably more suitable for building one's own concept network and theory than studying the specifics of someone else's conceptual network. Besides, map analysis as delineated by Carley (1993) and Carley/Palmquist (1992) takes a quantitative approach, which aggregates concepts and relationships; this is not supported by ATLAS.ti.

3. Exporting data

Most qualitative programs maintain a system file encompassing either some or all types of information of a specific text analysis project, e.g. the original text data, the coded text segments, the categories, the case variables or the socio-demographic data, etc. Some programs support the merging of their system files, e.g. ATLAS.ti, NUD*IST, and WinMAXpro. Merging refers to building one project from a number of separate projects created within the same software tool, its main purpose being to support teamwork. Merging is a package-specific operation: none of these files may be used with a different package.

Over half of the programs reviewed provide to a greater or lesser extent functionality for further processing with a statistics package. Rather than frequency of occurrence information about the words of the source text data, information which is to be processed statistically concerns the occurrence or non-occurrence or the exact frequency of occurrence of categories in relation to a text or a text unit, such as paragraph, speaker, subcorpus, etc. Most of the programs which support the export of coding provide an SPSS syntax and data file for processing the information about both the occurrence and frequency of occurrence of codes and coded text segments with the statistical package SPSS. WinMAXpro and WordStat store such information in a dBase file which can also be processed by SPSS. As a module of the statistics package SIMSTAT, WordStat supports the direct statistical processing of the coded data. Other programs store the coded data in ASCII files or generate project reports which include information about the frequency of occurrence of codes and coded text segments. Such information may be input to a statistics package provided that the user edits the texts appropriately afterwards as well as defines the appropriate variables and labels.

Information about the coding is stored

- as specific kinds of scores, e.g. DIMAP-MCCA or DICTION
- in the form of whether a code is assigned or not assigned to a specified text unit
- as a sequence of codes for a text unit
- as frequency of occurrence of codes for a text unit
- as time-series data, e.g. KEDS.

The definition of the text unit varies according to each program. For example, TEXTPACK and CoAn use structural elements of a text (called IDs), for TextSmart it is the whole text itself that is an answer to an open-ended question, ATLAS.ti and TATOE use paragraphs, WinMAXpro uses coded text segments. WinMAXpro and ATLAS.ti in addition specify the start and end position of the coding in the text element.

Similar to our discussion on importing text data, there is little support for exporting the text analysis data in a format which will enable other researchers with similar text analysis purposes and using different packages to use the same data for analysis. This is in regards to all different types of information, namely, the export of the text data without coding, the coding itself, the coding scheme as well as all or a specific combination of the above information types. The dissemination in a standard format which would guarantee that no pre-editing is necessary and no loss of information of both raw and coded text data occurs is hardly supported by the software reviewed. As a solution to this problem the developers of AQUAD, INTEXT, and WinMAXpro agreed on a data exchange format for text analysis (see Kuckartz 1998). This means that the coding performed in AQUAD, WinMAXpro or INTEXT can be saved in a way that makes the coded data exchangeable within these programs.

For the presentation of the analysis data CoAn and ATLAS.ti support the generation of HTML-formatted reports with hyper-linking between the different information types, e.g. of the coded text and the categorisation scheme. However, no support is provided for text data export for exchange and re-usability purposes. New developments in this area are the ATLAS.ti support for exporting memos to XML and the functionality provided by TATOE for exporting text and coding to XML.

It appears that there is no sufficient support for either

- exploiting already existing data or
- validating the analysis performed by other researchers or
- performing secondary analysis using the same text material but with a different text analysis package or with other software.

As a way round this problem it would be worthwhile to examine the extent to which the work of the Text Encoding Initiative (TEI) guidelines on encoding and interchanging machine readable texts by means of using the ISO standard SGML (Sperberg-McQueen/ Burnard 1994) is useful and relevant. Alexa/Schmidt (1999) describe the SGML modelling of answers to open-ended questions as raw, coded, and archived texts in an TEI-conformant manner and the advantages of such encoding for text analysis purposes.

Data export and the issue of dissemination and exchangeability concerns not only the text material but also the re-usability of coding schemes and dictionaries developed and used for text analysis. As far as dictionaries are concerned, as already mentioned in the previous section, text analysis programs differ with regards to whether the dictionaries used are system-internal or external. System-internal dictionaries have been developed for the purposes of the particular variant of analysis a program supports, for example for calculating the E- and C-scores when using DIMAP-MCCA. As such the dictionaries may not be exported or re-used with other text analysis programs. The dictionaries used by KEDS or those used for automatic analysis with CoAn, TEXTPACK and WordStat are all text files stored external to the respective programs. Such dictionaries may be edited and to a greater or lesser degree be re-used for other analysis projects. The format and structure of information for each dictionary entry is, however, not the same and therefore pre-editing is often necessary.

TextSmart saves the Exclusion and Alias word lists as ASCII text files, but merely those aliases created by the user and not those automatically created by lemmatisation are included. Categories in TextSmart cannot be exported or re-used in other projects because the categories are directly based on the Inclusion word list which is generated for each text on basis of the words in a text.

With regards to categorisation schemes, text analysis programs output all categories created as an ASCII file. The types and amount of information contained in this file as well as the way this is structured differ. The user of ATLAS.ti has the option to output the created codes either as a list or as a hierarchy (remember that ATLAS.ti supports the definition of different kinds of relations holding between two codes). The categories *Magic*, *Black Magic* and *Kabbala* (taken from the ATLAS.ti demo) are listed as follows if the former option is chosen:

```
Magic
Black Magic
Kabbala
```

and as follows if the latter option is chosen:

```
Magic  <is>  Root
  Black Magic  <is a>  Magic
  Kabbala  <is associated with>  Magic
```

The hierarchy option provides information about the relations holding between the categories, super-categories are roots and tabs are used to indicate the hierarchy. The user of WinMAXpro can also output the codes to an ASCII file and the information appears as follows (taken from the demo version of WinMAXpro):

```
Private issues [0:0]
   family priorities [4:17]
   friends [7:38]
Public issues [0:0]
   drugs legal [1:3]
   education [5:6]
      education policy [2:5]
   Law and order [2:10]
   The press [1:1]
      Murdoch [3:3]
   transport [1:3]
   welfare state [0:0]
```

Here tabs are also used to indicate sub- and sub-sub-categories. Information about the amount of times a category has been assigned and the total of text lines coded according to each category is provided and displayed encoded in square brackets. If the user wishes to import the above coding scheme as output from WinMAXpro in ATLAS.ti or TATOE, then the hierarchy information will be lost, since both programs support the import of merely a list of codes and not of a hierarchy of codes. Moreover, the user must pre-edit the file and delete the frequency information before import. On the other hand, if one would like to import the code list as output by ATLAS.ti in WinMAXpro, one must either pre-edit the file or the file must be in dBase format for WinMAXpro to import it accordingly. Information about the relations holding between the categories cannot be imported and will therefore be lost.

Obviously, standardisation of both the information elements a coding scheme involves and their format would enhance the exchangeability and re-usability of dictionaries and coding schemes as well as keep the analysis material independent from the text analysis package one uses.

4. Linguistic processing

KEDS, DIMAP and partly TextSmart involve linguistic processing and incorporate linguistic information. Although not a full syntactic parser for English, KEDS uses pattern recognition to parse specific elements of a news lead sentence, such as verbs, objects of a verb phrase and proper nouns. The KEDS parsing mechanism is dependent on semantic domain and text-type restrictions. KEDS focuses not only on the identification of types of concepts in a text, but also on identifying and explicitly encoding the connections between the concepts of a news lead sentence. The program uses three primary types of information which are stored in separate files for verbs, actors, i.e. proper nouns referring to political actors, and phrases. In principle, these files store the necessary 'knowledge' KEDS requires in order to both distinguish between the different meanings of a verb and locate and identify the sources and targets of an event within a sentence.

The dictionary maintenance program DIMAP supports the definition of both different senses of words of a dictionary, keeping in this way homographs separate and semantic restrictions encoded in word senses. DIMAP also provides functionality for specifying hypernym and hyponym links, attribute-value feature structures, and semantic links to create semantic networks. Unfortunately, the advantages of combining a dictionary maintenance program with a dictionary-based content analysis program such as MCCA cannot yet be demonstrated in practice as "the real integration (where the resources of DIMAP can be used to extend the dictionary used in MCCA) has not yet occurred. There are several theoretical issues involved, primarily the question of how to perform automatic re-weighting."[21]

Some of the programs reviewed which code text automatically based on a dictionary use lemmatisation, stemming, part of speech categorisation and word sense disambiguation techniques. In stemming the program matches the beginning of different forms of (possibly) the same word, e.g. *say*, *saying*, and *says*, to a string called stem. Lemmatisation, in contrast, refers to the matching of all different forms of a word regardless of whether its root is the same, e.g. *say* as well as *said* have all the same lemma.

TextSmart is the only software in our review which has an integrated automatic lemmatisation for English, but only for the 'most' frequent words[22]. When a text file is imported as well as every time the file is edited and updated, TextSmart automatically lemmatises the text data and creates a word alias, i.e. a group of different word forms identified by the lemma. Stemming may be used by other programs, e.g. CoAn, KEDS, TEXTPACK, with

---

21 Ken Litkowski, June 1998, personal communication.

22 We have found no pointer to the exact threshold for a word to belong to the most frequent words.

the stems as dictionary entries delimited by a special character. This means that the user specifies in advance which words are to be treated as stems. WordStat uses its Substitution Dictionary which, in principle, contains the categories of the coding, but may also be used for defining lemmata, for example, the word forms *art*, *arts* or even *artistic* are related to the category 'art'.

KEDS makes use of part of speech categorisation, since the data files it uses for coding are organised into groups of verbs and proper nouns or nouns. When one of the reviewed text analysis packages uses part of speech information, this is only partial: not all words of the text(s) analysed are categorised for part of speech. Rather specific linguistic information, such as categorisation in verbs or nouns, serves as a means for further categorisation. In other words, part of speech tagging is not part of the programs' functionality. For those programs which use part of speech information, as far as we have been able to check from the available literature, there are no pointers to or information about whether part of speech categorisation has been performed manually or automatically during the dictionary construction. Another way of exploiting part of speech information is as this is supported in TATOE: German texts which have been automatically analysed by means of the Morphy tagging program may be imported in TATOE as coded texts, and the part of speech categories are stored as a separate categorisation scheme.

DIMAP-MCCA and DICTION apply word disambiguation procedures based on frequency of occurrence information. In particular, MCCA uses context information and the expected frequency of occurrence within a given context in order to disambiguate a word, whereas DICTION uses frequency information which is based on counts of homographs differentially weighted within the various dictionaries by applying (a priori) statistical norms for their use. Unfortunately, we have not found references on tests made to measure the accuracy of disambiguation procedures of both of these programs. Systematic evaluation of the content analysis disambiguation or comparisons of the disambiguation techniques in content analysis with more linguistically oriented disambiguation ones are both areas for future work.

The recognition of negation in order to exclude negated text segments from coding or – when necessary – to code them accordingly is poorly supported by programs with an automatic coding procedure based on a dictionary. For instance, a few of the DIMAP-MCCA dictionary categories are slanted toward negative words. INTEXT (Klein 1997b), which was not selected for the review (INTEXT is a DOS-based program), uses lists of words, stems or word suffixes which indicate negation expressions, e.g. *not*, *no*, *never*, *un-*, which the user constructs to deal with the recognition of negation.

The majority of the programs hardly go beyond the text; the meta-textual information included in headers and the factual, case- or project-related information. Lexical, syntactic or semantic information, such as the WordNet (Miller et al. 1993) semantic database for English, thesauri or ontologies, or other additional text sources (text corpora), is but rarely made use of or coupled with the analysis. An exception to this is the work reported by McTavish et al. (1997) on using WordNet for examining the categories of the MCCA dictionary: McTavish et al. show that the MCCA dictionary categories are consistent with the WordNet synsets (sets of synonyms connected with each other with a number of semantic relations, e.g. part-of, hyponymy, antonymy, etc.). In particular, the usage of the WordNet data is shown to be advantageous in two main ways:

- it allows the analyst to move from subjective conceptual or contextual groups to more general semantic groups which reflect fine-grained meanings inherent in particular words and
- it enables the further refinement of the content categories into semantic components: it is possible to extend the amount of words that might be associated to a category, mainly by identifying and including the narrower terms (hyponyms).

Both of these may result in explicit and transparent analysis. In general, enriching text analysis with such 'external' linguistic information may have advantages for the exploration of texts, the construction of categorisation schemes and dictionaries, and the testing of their validation. It may also contribute in providing more fine-grained characterisation of concepts and themes and in analysis based on more rigorously defined principles.

It may be worthwhile to opt for more modularity in order to expand the applicability of the programs employing language-dependent linguistic information and techniques for analysing texts in other languages: the language-dependent processes of a program are separate modules and the program offers documented interfaces to these. This would also accommodate the possibility for other researchers to develop and integrate their own routines, e.g. as DLLs, for the specific language and grammar-dependent parts of the program.

5. Exploring text data and coding

Exploration refers to the functionality of text analysis programs for searching and retrieving texts, words or strings of words, single codes or combinations of codes, and memos. Programs vary with respect to whether their support for exploration focuses on exploring the text or coding. Qualitative-oriented programs appear to place more emphasis on exploring coding with a number of them supporting complex and advanced queries; for example, for examining co-occurrences or juxtapositions of codes. In contrast, programs whose focus is on the automated coding of texts support mainly general text searches for words or word strings or queries with Boolean, i.e. AND, NOT, OR operators, display the retrieved occurrences in KWIC lists, generate a word index of all words of the text(s) analysed, and provide frequency of occurrence information. Their support for the exploration of existing coding is fairly rudimentary in comparison to qualitative-oriented programs.

We have discussed in the second part of this book the advanced techniques for searching and exploring coding of the QueryTool of ATLAS.ti (see Part 2, section 2.5), the Index Search operators (see Part 2, section 9.5) of NUD*IST, the Logic Machine facility of WinMAXpro (see Part 2, section 14.5) or the Test Hypothesis functionality (see Part 2, section 7.5) of HyperRESEARCH for defining conditional expressions. Such support for querying and examining the coding is hardly provided by quantitative-oriented programs. The assumption which appears to be made as far as the latter are concerned is that if the coding results can be exported to a statistics package then coding can be further explored there. However, better possibilities to explore and check coding are important kinds of support a text analysis program could provide for the processes of both building and validating categorisation schemes and dictionaries.

For the purposes of developing both coding schemes and dictionaries as well as for examining their validity for the research question one is analysing the texts for, it is invaluable to obtain co-occurrence information of either words or strings of words or codes. For instance, it is often helpful to be able to list all words (or stems) which occur either to the left or to the right of a specified word or a string of words or a code, or to both left and right side, within the same ‘text unit’; the size of the text unit (e.g. sentence, a paragraph or the whole text) for calculating co-occurrences should not be hard-coded and given, but rather the user could determine each time what the unit should be. It must be noted that such functionality is dependent on the degree of text structuring supported by a text analysis software.

A feature not supported by the majority of the programs we examined is the possibility to define searches which include sequences or alternations of codes *and* strings. Exceptions

are Code-A-Text, NUD*IST and TATOE, as well as ATLAS.ti with the restriction that only the Boolean operator OR can be used for such searches.

The matches to a query for the occurrences of a word, phrase, code or combination of codes, can be displayed as a KWIC list where the left and right context for each match is provided. Some of the reviewed programs do not provide KWIC displays, e.g. HyperRESEARCH, KEDS or QED. Programs which do realise this in a variety of ways. WinMAXpro will display the result of a text search as a KWIC display, with the restriction that the user can only view this off-line: the retrieved instances are saved as an ASCII text file. CoAn, DIMAP-MCCA or TEXTPACK provide KWIC displays but the left and right context is hard-coded and can neither be increased nor decreased by the user. A further distinction concerns whether KWIC displays are tied to either text exploration or coding exploration only, or whether they are provided irrespective of whether the query performed concerns a word or a coded text segment. For example, CoAn will generate a KWIC display, but only for a word or a phrase contained in the dictionary and not for any word or a string of words occurring in the text data. In contrast, the KWIC output of WinMAXpro results after a text search and not after a search for the occurrences of a particular code. TATOE and WordStat, on the other hand, will provide KWIC lists not only for words or strings of words but also for codes. Furthermore, contrary to TATOE, the KWIC displays of CoAn, TEXTPACK, and WinMAXpro are not interactive: the user can view, print or edit the KWIC display with the retrieved matches, but since there is no linking between the system and such a display, she cannot select further a word or a string of words from that list and generate a further KWIC list as is supported by TATOE.

A different kind of support for exploration concerns the possibility to keep a record of searches and their respective matches which can be repeatedly selected at any time during analysis. One can furthermore distinguish between storing the rules for searches or keeping a logbook of the results of the searches. For example, search categories (called also search swarms) in ATLAS.ti may be stored under descriptive names and managed in search libraries. NUD*IST, on the other hand, makes a new node with the result of a search, attaching it to the nodes for Text Searches which in a way form a search library. The user may edit and/or save the result of the search under this node and browse it again at a later stage, or add it to the Index Tree and thus use it for coding. Results of any text or index search of NUD*IST may be used as input to further index system searches, and in this way one can build up complex searches. TATOE stores all performed searches as Occurrence Lists and at any time during analysis the user may select an item from the list and ask for a concordance display of the search matches. Occurrence Lists may not only be repeatedly selected but also be used to build up groups of different Occurrence Lists by means of addition or subtraction operations.

The advantages of storing the results are not only that matches may be repeatedly displayed without having to define and initiate the search each time, but also that operations can be performed on stored items, e.g. the matches may be coded, memos may be attached to them, or they be hyper-linked to other results, memos or categories.

Last but not least, an important feature for exploration purposes concerns the 'hyper-linking functionality' of the reviewed programs. In principle, we may view the code-and-retrieve functions of programs such as WinMAXpro, ATLAS.ti, QED or TATOE as hypertext functionality: links may be exist among

- categories within a single text or a number of texts
- categories themselves, since relations may be defined to hold among them
- texts, codes, coded text segments to (pop-up) comments, notes or memos
- categories to families, groups, parallel schemes, e.g. ATLAS.ti, TATOE.

Added to the above are graphs of hypertext-like networks (e.g. ATLAS.ti), lists or indexes of the words of a text or a text corpus, where on selecting a word the program displays it immediately as it occurs in the text (e.g. TATOE) or means for investigating the relationships between words, categories and responses (e.g. TextSmart's 'brushing' facility which when switched on broadcasts the specific selection across all the other windows each time the user clicks on a category or a text or an term). All these different kinds of linking mechanisms are means for navigating from one type of link to the next and assist in exploring the text and its coding.

6. User interface

Over half of the programs we have reviewed may be described as 'second generation' text analysis software with a simple Windows-look, which make little use of modern graphical user interface technology. More crucially, the interactivity and hyper-linking of the different windows with displays of the raw or coded text data, codes and coded instances, the matches to the search-and-retrieve operation, etc., are either only very basic or not supported at all. Often the user runs batch routines organised in menus. ATLAS.ti, QED, TextSmart, TATOE or WinMAXpro are exceptions to this.

Most of the reviewed text analysis programs often go from performing an operation directly to generating an output file as the result of this operation instead of incorporating an intermediate phase for browsing as a means of *working with* the results. An output which is a hardcopy-like list of all hits after a search operation, which contains either line numbers indicating the position of each hit or where each hit is displayed as a KWIC item whose left and right context of a word is hard-coded, may perhaps be appropriate for

saving it as a text file, but it is not necessarily fit for working with the information it contains. Future development needs to offer more flexible ways of browsing and manipulating text displays and retrieved occurrences: further on-screen selections, adjustment of the amount of context displayed, varied displays with coding switched on or off, etc. Displays of the text data and their coding as well as displays of the output of different kinds of searches have the purpose of providing different views on the data and keeping the user close to the text data. The generation of 'lump-it-all-in-one' reports is one of the possible ways of storing analysis results but not *the* best way for displaying the results on screen and working with them. During text analysis the user is considerably supported if she can see her text and its codes clearly and easily on screen, have direct access on or point-and-click text segments and perform varying operations. Programs such as AQUAD, NUD*IST or DICTION keep the user at a distance from her text during analysis.

An important aid to analysis is to be able to have different views of the data, i.e. raw text, codes, coded segments, etc., side-by-side and still have the possibility to perform further operations on them, for example, juxtaposing different kinds of information. A few programs, e.g. ATLAS.ti, TATOE, TextSmart or WinMAXpro, provide for the integration of different display means: usage of margins or highlighting and usage of different colours to display information about coded text segments, hypertext links, icons, graphical, e.g. tree and network displays, etc. Future text analysis software should invest more effort in both GUI design using various display means and modalities as well as their testing and evaluation.

7. Integrating modes of and methods for analysis

As stated by Tesch (1990: 28-29) on her review of qualitative software " [...] any given concrete study might well incorporate aspects from more than one research type. [...] the specific research purpose and nature of the data can differ from study to study even within one approach, necessitating unique data management manoeuvres." Spelling out some "cut-across needs and hopes" for qualitative analysis software, Weitzman/Miles (1995: 334) affirm that "there is a growing agreement in the qualitative research community that linking qualitative and quantitative data in study designs and analyses can enhance validity, develop richer analyses, and lead to deeper theoretical insight (...)." Furthermore, Stone (1997: 50) argues that "[T]echnology no longer constrains researchers to one or another mode of analysis. Instead it is feasible to consider a platform of desktop computer text analysis capabilities, that allows the researchers to draw upon sequences of procedures that best suit their proclivities, to follow up on leads as they uncover textual patterns, and to check the validity of their discoveries."

Incorporating aspects of different research types, linking of qualitative and quantitative data, and integration of different modes of analysis are not only non-controversial, but rather valid methodological decisions for text analysis projects. However, few of the reviewed software packages facilitate integration or linking of text analysis methods and modes of coding.

With regards to the mode of coding a software supports, few programs aid the user to code both on-screen (manually) and automatically. Automatic coding means that the categories/codes are assigned to text segments by the computer mainly on basis of a word list or dictionary (or a number of them). In addition to a dictionary, rules organised in a grammar may be used, as for instance KEDS does. Only some of the programs aiming at the automatic coding of text data also provide support of interactive coding: KEDS and CoAn control the automatic coding process and intervene if necessary, whereas the user of TextSmart or WordStat or Code-A-Text can select a word or a text segment and assign manually codes to it independent of the automatic coding process.

In contrast to programs which code text automatically on the basis of a dictionary with an optional set of rules, different types of automatic coding are supported by programs which focus on supporting the user to code text interactively: For example, by means of using the Auto Coding facility of ATLAS.ti the user can code either a string matching exactly the search string, or a word or a line or a sentence or a paragraph or the whole text which has been retrieved for a single Primary Document, i.e. the one the user displays at the time, or for a family of or all Primary Documents of a Hermeneutic Unit according a selected code thus combining text search and coding. The option of WinMAXpro for automatic coding concerns the coding of all lines where a search string occurs in a single step. The user of QED may assign codes to a subset of texts, as opposed to text lines, which have been retrieved after a search. NUD*IST keeps a log of searches and retrieved results, and the user may create a node, i.e. a category for the particular search, which means that all the text units retrieved are marked according to the new node.

8. Generality of use

It is fair to say that all the programs reviewed here have been designed and developed with concrete application projects and types of research in mind and, as it may be expected, there is no package whose generality of use can cover all kinds of text analysis and their respective operations. However, one can distinguish degrees of generality of use: The functionality of the majority of the reviewed programs for the automated coding of text is often restricted to the analysis of one particular text type. For example, TextSmart is designed for the analysis of survey texts only, i.e. answers to open-ended questions, or KEDS efficiency and functionality lies in its exploitation of a single text type, namely that of event reporting in news leads.

In general, software packages differ according to whether they are characterised by a survey analysis orientation or not: survey-oriented ones are designed to handle relatively short texts, code them mainly automatically and export the coding to a statistics package. A further distinction may also be made between programs whose strength lies more in the analysis for dialogue texts, e.g. Code-A-Text, and those programs which support the analysis of a variety of text types, e.g. ATLAS.ti.

The generality of usage of the text analysis programs is not independent of the type of categorisation a program assumes or supports. Programs whose categorisations are built-in and whose processing involves calculating scores for pre-defined, theory- and domain-bound categories have a narrower applicability spectrum in comparison to programs which either support the user in developing a coding scheme or use a system-external dictionary. DIMAP-MCCA belongs to the former: the MCCA dictionary is used in order to produce on the one hand scores for measuring social context according to four parameters, i.e. Traditional, Practical, Emotional, and Analytic, and on the other hand individual emphasis scores for 116 categories. These categories were developed judgementally and are considered important for the analysis of naturally occurring verbal material.

DICTION is another text analysis program with a built-in dictionary: the program is designed specifically for political discourse analysis and it measures the style of political texts according to the ‘general’ categories Activity, Certainty, Realism, Optimism, and Commonality based on individual scores for twenty-five sub-categories which are considered to be important for the interpretation of this kind of texts.

The text to be analysed comes, of course, in a particular language using a specific set of alphabet characters. The degree of applicability of package for the analysis of a text in a particular language depends on whether the package can process and display the alphabet of the text. Software which does not support the display and the processing, e.g. alpha-

betic sorting, stemming, etc., of texts in non-Latin based languages, using other alphabets and characters, naturally can not be used for projects which either concentrate wholly on text material in other than Latin based languages, e.g. Hebrew, modern or ancient Greek, Japanese, etc., or use texts from different languages, for example, Russian and American English reports of the same event, Swedish, Norwegian, and German texts which are translations of each other, etc.

Concluding remarks

Considering the variety and multitude of text analysis programs and their respective functionality one cannot but be positively impressed and excited; especially with regards to the possibilities opened for interesting and creative analyses as well as comparisons of analyses and methods. We believe that depending on its degree of complexity, the functionality offered by each of the reviewed software may cover fully the needs of a given text analysis project.

By no means do we wish to suggest in this review that one should aim at developing a single 'ideal' software for all kinds of text analysis and all projects. Nor do we propose that text analysis operations should be uniform. We do expect, however, that some agreement on or development of standards (text encoding, export formats, etc.), as well as some agreement on standard terminology for basic concepts in text analysis software would be an improvement. Furthermore, we hope that the review contributes to the determination of the more or less 'core' or 'basic' functionality that a text analysis program worth its money and the time invested to learn to use it should offer.

Our aims have been, firstly, to provide both a detailed and an up-to-date account of the spectrum of text analysis software that aid, to a greater or lesser extent, coding and catalogue the kinds of support these offer to the user and, secondly, to record the tendencies in functionality and technology and identify the areas where more development is needed. We are aware that there might be candidate programs which, given their functionality, should have been included in this review but which are not. Some possible reasons for this are insufficient information on our part or 'timing': for example, on the very day we were rounding up the second version of the review we saw that the Windows version of Ethnograph (see http://www.scolari.com/ethnograph/scolari_ethnograph.htm) was available. As there was a deadline for this review, Ethnograph could not included here. The same holds for the NVivo software developed by the NUD*IST developers which was announced in February 1999.

We hope that this review will not only assist text analysts in choosing between available programs, but also in questioning, testing, and enriching their analysis methodology.

Furthermore, we hope that this review will contribute to the dialogue between text analysts adopting different text analysis approaches in different disciplines and developers of software for text analysis.

Finally, and as an afterword, throughout the process of reviewing text analysis software it has become clear to us that the review of a software should not be equated with applying and working with it for a particular project. The process of examining a software in order to provide a review of it centres, among other things, around

- determining the kind of functionality provided,
- examining how specific things work and
- estimating how well specific tools or options support particular tasks.

What also must be examined is what consequences certain kinds of functionality or ways of performing text analysis tasks have for the analysis itself. However, if there is no particular text analysis project or question, specific features may escape the reviewer's attention or the importance of a specific type of functionality may not be assessed appropriately. When analysts with a particular research question use a program for their text analysis, different features of a program may appear more important than others. Furthermore, one should also take into account that analysts are often creative in using some of the program's functions for purposes other than the ones the developers of the program have designed them for or the reviewers have understood them to be there for.

Appendix

AQUAD

http://www.aquad.com
Guenter Huber, University of Tuebingen, Germany.

Order from:

Ingeborg Huber Verlag, Postfach 46, 87643 Schwangau, Germany
Tel.: +49 (0)8362 987073
Fax: +49 (0)8362 987073
e-mail: Guenter.L.Huber@t-online.de

Price: $190.00 (academic, 1 user)

ATLAS/ti

http://www.atlasti.de/
Thomas Muhr, Berlin, Germany.

Order from:

Europe:
SCOLARI, Sage Publications Ltd, 6 Bonhill Street, London, EC2A 4PU, UK
Tel.: +44(0)171 330 1222
Fax: +44(0)171 374 8741
North, Central and South America:
SCOLARI, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks,
CA 91320, United States
Tel.: + 805 499 1325
Fax: + 805 499 0871

Price: $395.- (academic, 1 user)

CoAn

http://gwdu19.gwdg.de/~mromppe/soft/soft.htm
Matthias Romppel, University of Goettingen, Germany.

Price: Beta-test version. Freely-available

Code-A-Text

http://www.codeatext.u-net.com/
Alan Cartwright, University of Kent at Canterbury, UK.

Order from:

Europe:
SCOLARI, Sage Publications Ltd, 6 Bonhill Street, London, EC2A 4PU, UK
Tel.: +44(0)171 330 1222
Fax: +44(0)171 374 8741
North, Central and South America:
SCOLARI, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks,
CA 91320, United States
Tel.: + 805 499 1325
Fax: + 805 499 0871

Price: $315.- (academic, 1 user)

DICTION

http://www.scolari.com/diction/Diction.htm
Roderick Hart, University of Texas, USA.

Order from:

Europe:
SCOLARI, Sage Publications Ltd, 6 Bonhill Street, London, EC2A 4PU, UK
Tel.: +44(0)171 330 1222
Fax: +44(0)171 374 8741
North, Central and South America:
SCOLARI, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks,
CA 91320, United States
Tel.: + 805 499 1325
Fax: + 805 499 0871

Price: $129.- (academic, 1 user)

DIMAP-MCCA

http://www.clres.com/
Ken Litkowski, CL Research, Gaithersburg, MD, USA.
Don McTavish, University of Minnesota, MN, USA.

Order from:

CL Research, 9208 Gue Road, Damascus, MD 20872-1025
Phone: 301-482-0237
e-mail: ken@clres.com

Price: $200.- (DIMAP 2.0) (academic, 1 user), beta test version free.

HyperRESEARCH

http://www.researchware.com/
ResearchWare, Randolph, MA, USA.

Order from:

ResearchWare, Inc., P.O. Box 1258, Randolph MA 02368-1258
Phone & Fax: 781-961-3909 (US)

Price: $225.- (academic, 1 user)

KEDS

http://www.ukans.edu/~keds/
Philip Schrodt, University of Kansas, USA.

Order from:

KEDS Project, Department of Political Science, University of Kansas,
Lawrence, KS 66045 USA
Tel.: 913-864-3523
Fax: 913-864-5700
e-mail: p-schrodt@ukans.edu

Price: Beta-version. Freely-available

NUD*IST

http://www.qsr.com.au/software/n4/n4.htm
QSR Software, La Trobe University, Australia.

Order from:

Europe:
SCOLARI, Sage Publications Ltd, 6 Bonhill Street, London, EC2A 4PU, UK
Tel.: +44(0)171 330 1222
Fax: +44(0)171 374 8741
North, Central and South America:
SCOLARI, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks,
CA 91320, United States
Tel.: + 805 499 1325
Fax: + 805 499 0871
Australia, New Zealand:
Qualitative Solutions & Research Pty Ltd, Box 171 La Trobe University Post Office,
Victoria 3083, Australia
Tel.: +61(0)3 9459 1699
Fax +61(0)3 9459 0435
e-mail: nudist@qsr.com.au

Price: $373.- (academic, 1 user)

QED

http://www.trimm.nl/qed/
TriMM MultiMedia, Holland.

Order from:

MiMiC, P.O. Box 1208, 7500 BE Enschede, The Netherlands
fax: +31 53 4353027
e-mail: support@qed.mimic.nl

Price: NFL 375.- (academic, 1 user)

TATOE

http://www.darmstadt.gmd.de/~rostek/tatoe.htm
Melina Alexa, ZUMA, Mannheim
Lothar Rostek, GMD-IPSI, Darmstadt, Germany.

Price: Beta-test version. Freely-available

TEXTPACK

http://www.zuma-mannheim.de/software/en/textpack/
ZUMA, Mannheim, Germany.

Order from:

ZUMA, P.O. Box 12 21 55, D-68072 Mannheim, Germany
Tel.: +49 621 1246147
Fax: +49 621 1246100

Price: DM 200.- (academic, 1 user)

TextSmart

http://www.spss.com/software/textsmart/
SPSS Inc., Chicago, USA.

Order from:

SPSS Inc., 233 S. Wacker Drive, 11th floor, Chicago, IL 60606-6307
Tel.: (800) 521-1337
Fax: (800) 841-0064
and all local SPSS offices in Europe

Price: DM 1500-1800,00 (academic, 1 user)

WinMAXPro

http://www.winmax.de/
Udo Kuckartz, Berlin, Germany

Order from:

Germany:
BSS Büro für Softwareentwicklung und Sozialforschung, Schützallee 52,
D-14169 Berlin
Tel/Fax: +49 30 8137201

Rest Europe:
SCOLARI, Sage Publications Ltd, 6 Bonhill Street, London, EC2A 4PU, UK
Tel.: +44(0)171 330 1222
Fax: +44(0)171 374 8741
North, Central and South America:
SCOLARI, Sage Publications, Inc., 2455 Teller Road, Thousand Oaks,
CA 91320, United States
Tel.: + 805 499 1325
Fax: + 805 499 0871

Price: DM 550.- (standard), DM 750.- (WinMax professional),
Scolari: $350.- (academic, 1 user)

WordStat

http://www.simstat.com/
Normand Peladeau, Provalis Research, Montreal, Canada.

Order from:

Provalis Research, 2414 Bennett Street, Montreal, QC, CANADA, H1V 3S4
Tel.: (514) 899-1672
Fax: (514) 899-1750

Price: SimStat ($129.-), WordStat ($59.-). (academic, 1 user)

Glossary

This glossary summarises alphabetically all terms included in the Terminology sections of each reviewed program.

Activating Text: (WinMAXpro)
The user can control which coded text segments should be displayed by WinMAXpro. The user may activate one, more than one or all texts. Activating a text is necessary also for performing all operations related to a text, e.g. searching, obtaining frequency of occurrence information, etc.

Active Words: (Code-A-Text)
Words defined by the user as relevant and meaningful for the purposes of analysis.

Alias List: (TextSmart)
Contains a list of related words which are all considered equivalent and should be thought of as groups of synonyms. Typically, two kinds of alias lists are built: semantically-related and morphologically-related (failed to be joined during stemming).

Analysis card: (QED)
Similar to a memo, it may be used to keep track of ideas and observations.

Archive: (Code-A-Text)
It contains a dictionary of all the words known to the project, a list of all the Active Words and a list of the Word Groups.

Brushing: (TextSmart)
A TextSmart feature which helps to investigate the relationships between words, categories, and responses. When brushing is 'on', each time the user clicks on a category or a text or an Included Term or Excluded Term the selection is broadcasted across all the other windows.

C-score: (DIMAP-MCCA)
Stands for context-score. It is one of the two types of the scores provided by MCCA which consists of a profile of four social contexts scores: traditional, practical, emotional, and analytic.

Case: (HyperRESEARCH)
The basic unit of analysis in a HyperRESEARCH Study. It contains the codes and the coded text segments for one or more text documents.

Categorical Scales codes: (Code-A-Text)
User-defined codes which contain a set of non-numeric values.

Category System: (CoAn)
Dictionary on the basis of which automatic coding can happen. It contains the category numbers, and the search expressions and may contain parameters (e.g. whether upper-lower case should be considered or not).

Coding Frame: (Code-A-Text)
A group of database tables which store the project information like the Coding Manual and the texts to be analysed or which have been coded.

Coding Manual: (Code-A-Text)
A 'database' storing speaker identifiers, names of valid scales, and code names for each scale.

Command File: (NUD*IST)
Similar to scripts or macros, a command file can automate almost any aspect of the NUD*IST work.

Composite pattern: (KEDS)
A composite pattern contains a set of alternative matches.

Content code: (Code-A-Text)
An Active Word whose occurrences in text are coded.

Data Card: (QED)
Texts in QED are divided into 'meaningful' chunks and placed on Data Cards.

De-activated Text: (WinMAXpro)
The coded text segments of a de-activated text are not included in the window containing the list of coded text segments. One, several or all texts, as well as one or more codes must be activated/de-activated in order to control which coded text segments will be displayed in the respective window which lists the coded segments.

Dictionaries: (DICTION)
Twenty-five word lists for each DICTION 'sub-category', e.g. 'Tenacity', 'Collectives', 'Numerical Terms'.

E-score: (DIMAP-MCCA)
E-score stands for emphasis score. One of the two kinds of scores that are the result of the MCCA processing, which represents the emphasis placed on each of many idea categories.

Excluded Terms: (TextSmart)
Words or aliases that have 'little semantic value' (as opposed to 'content words') and cannot be used to build categories.

Exclusion Dictionary: (WordStat)
A list of words which are not considered for the analysis; the so-called stop word list.

External Document: (NUD*IST)
Any document which can be coded without being imported.

Family: (ATLAS.ti)
Different object types, i.e. Primary Documents, codes, or memos can be grouped in families. One code or Primary Document or memo may belong to more than one family.

Free Node: (NUD*IST)
A category which has not been organised in the Index tree.

GO Words: (TEXTPACK)
Words which are considered for the analysis.

Hermeneutic Unit: (ATLAS.ti)
The 'container' for an analysis project. Primary Documents are assigned to a Hermeneutic Unit, which is the most prominent data structure the user is working with.

Idea category: (DIMAP-MCCA)
An idea category onsists of a group of words that reflect a given idea or meaning. For example, the idea of 'control' occurs when words such as 'allow', 'authorize', 'insist', and 'command' are used. The MCCA dictionary distinguishes 116 idea categories.

Identifications (IDs): (TEXTPACK)
Markers for each text unit in a text corpus.

Implicant: (AQUAD)
A module that allows the user to perform Boolean qualitative comparisons of the data (to support investigation of causality).

Included Terms: (TextSmart)
Words or aliases that are used to build categories, either interactively by the user or automatically by TextSmart. The inclusion list is built by the program using all words in the text and the alias list without the terms in the excluded list.

Inclusion Dictionary: (WordStat)
A list of all the categories used for coding (note that the WordStat Help text uses the word "words" instead of the word categories).

Index Tree: (NUD*IST)
A hierarchical categorisation scheme.

Interpretation codes: (Code-A-Text)
User annotations, i.e. memos.

Linkages: (AQUAD)
Search algorithms for viewing how two or more codes 'link' or co-occur in the same document.

Logical Activation of Texts: (WinMAXpro)
It allows the user to use the values of existing variables in order to formulate logical conditions for searching and selecting, i.e. activating, only those texts for which the conditions hold.

Master Code list: (AQUAD)
A list of all codes used in all files of a project.

Memo (or Memo Field) and Record: (WordStat)
A text unit which accepts up to 32K of text. WordStat can analyse text stored in memo field (up to 32K) or alphanumeric fields (only 255 characters). A single record is usually associated with a respondent and can contain answers to different open-ended questions.

Meta-codes: (AQUAD)
Meta-codes are super-ordinate codes.

Nodes: (NUD*IST)
Nodes are primarily categories. As stated in the NUD*IST Handbook nodes are "containers for your thinking about the project and for your index references (coding) into the data document" (Richards 1998: 16). Nodes must have a title and an address and can contain a definition, coding or memo. However, the meaning of the word 'ideas' is stretched: on the one hand, nodes *do* represent ideas for analysis but, on the other hand, they can be variables for the sub-grouping of documents, for example 'female', 'under 30'.

Numeric Scale codes: (Code-A-Text)
Codes with numeric values. These are computed by the program taking into account the frequency of occurrence of particular words of a text segment.

Occurrence Lists: (TATOE)
A TATOE-Notebook page which contains a logbook of searches which have already been performed and their results.

Pattern Search: (NUD*IST)
This term is used for any search for a pattern of characters, rather than a simple string. E.g. to specify an (unordered) set of words, e.g. [air | you | healthy] . The string of characters specified may include special character options.

Preprocessor: (WinMAXpro)
A WinMAXpro option which helps to import numerous and pre-structured (pre-coded) texts.

Primary Documents: (ATLAS.ti)
The original documents to be analysed and coded.

Quotations: (ATLAS.ti)
Each text segment (a word, a group of words, a whole paragraph, etc.) which is assigned to a code forms a quotation.

Report Extrapolations: (DICTION)
An option for working with texts whose size is less than 500 words and which makes corrective counts for standardising them to a 500-word basis.

Search Categories or Search Swarms or Search Patterns: (ATLAS.ti)
Definitions of search strings which may be given a name. Another term used for this purpose is a definition of a text search.

Search Expressions: (CoAn)
Words, stems or strings of words contained in the Category System. A specific syntax is used for differentiating between these.

Search Pattern Definition: (TATOE)
A user-defined search pattern which may contain words, strings or categories or a combination of them, e.g. look for all occurrences of the word ‘party’ followed optionally by either the category ‘Values’ or the category ‘Political Systems’. Search pattern definitions may be stored as rules of a grammar which may be re-used for searching or coding purposes or be included in other search pattern definitions.

Set of Cards: (QED)
A cross-section of the cards in the data files which "answer to a set of characteristics", and which are included in a Workbook. The user specifies what these characteristics are. Sets can be build automatically and manually.

Source (1): (HyperRESEARCH)
An ASCII/ANSI text file. One or more Sources may be opened and coded for a single Case.

Source (2): (KEDS)
Refers to the actor of an event data record who initiated an action.

Sparse Sentence Parsing: (KEDS)
Refers to the analysis of particular parts of a sentence, e.g. identification of verbs, their subjects and objects, but not full-blown and detailed syntactic analysis.

Speech Unit: (Code-A-Text)
A text structure unit corresponding to dialogue turns which are designated by a special delimiter. A Speech Unit contains at least one Text Segment. For non-dialogue texts a paragraph is a Speech Unit.

STOP Word List: (TEXTPACK)
Words excluded from further analysis.

Study: (HyperRESEARCH)
The entire text analysis project. It contains the Cases, codes, and all references to Sources. A HyperRESEARCH Study consists of at least one Case.

Substitution Dictionary: (WordStat)
The dictionary categories and their entries. The categories included may not only group words denoting a 'semantic' category, but they may also be groups of word forms having the same lemma.

Super Code: (ATLAS.ti)
Different from normal codes in that it stores a query formed using other codes.

Target: (KEDS)
Refers to the actor in an event data record who is the object of an action.

TATOE-Notebook: (TATOE)
A separate window which is organised as a virtual 'notebook' whose 'pages' are dialogue windows for performing main operations in TATOE, e.g. creating a coding scheme, importing and exporting texts, etc.

Textbase: (TATOE)
An object network or a kind of database produced by importing the texts.

Text Group: (WinMAXpro)
A subset of the text data of a WinMAXpro project. At least one text group must be defined in order to work with WinMAXpro.

Text Segment: (Code-A-Text)
Is either a dialogue turn, in which sense it is a Speech Unit, or a smaller part of a speech unit, as defined by the user. A Text Segment is the coding unit in Code-A-Text, i.e. the unit to which one or more codes are assigned.

Text Units: (NUD*IST)
The smallest units of the document the user may code or retrieve, e.g. a page, a paragraph, a sentence, etc.

Word Groups: (Code-A-Text)
User-defined content categories which contain Active Words. For example, the Word Group ‘anxiety’ may consist of the following words: anxiety, avoidance, churn, churning, doubt, worried, worry and worrier.

Words, Multiple-word Combinations, and Word Roots: (TEXTPACK)
A word in TEXTPACK is a string of characters which is preceded and followed by one (or more) blanks. A Word Root is defined as the left part of a word. Multiple-word Combinations are defined as sequences of single words.

Workbook: (QED)
A system file which provides the framework for a QED analysis project. If the user wants a data file to be included in the set operations this has to be included in the Workbook. In other words, sets and set operations are only possible for Workbooks.

References

Alexa, M., 1997: Computer-assisted text analysis methodology in the social sciences. ZUMA Arbeitsbericht 97/07. Mannheim: ZUMA.

Alexa, M., 1998: Text type analysis with TATOE. Pp. 247-264 in A. Storrer/B. Harrienhausen (eds), Hypermedia für Lexikon und Grammatik. Tübingen: Gunter Narr Verlag.

Alexa, M./Schmidt, I., 1999: Modell einer mehrschichtigen Textannotation für die computerunterstützte Textanalyse. Pp. 323-345 in W. Moehr/I. Schmidt (eds), SGML/XML – Anwendungen und Perspektiven. Heidelberg: Springer Verlag.

Alexa, M./Rostek, L. 1996: Computer-assisted, corpus-based analysis text with TATOE. Pp. 11-17 in ALLC-ACH96, Book of Abstracts, Bergen, Norway.

Barry, Ch. A., 1998: Choosing qualitative data analysis software: Atlas/ti and Nudist compared. Sociological Research Online, Vol. 3, No. 3, http://www.socresonline.org.uk/socresonline/3/3/4.html.

Bazeley, P., 1998: NUD*IST Community Project Exercises. QSR.

Carley, K., 1993: Coding choices for textual analysis: a comparison of content analysis and map analysis. Sociological Methodology, 23: 75-126.

Carley, K./Palmquist, M., 1992: Extracting, representing, and analyzing mental models. Social Forces, 70: 601-636.

Depuis, A./Tornabene, E., 1993: HyperRESEARCH ™ from ResearchWare: a content analysis tool for the qualitative researcher. Randolph, MA: ResearchWare, Inc.

Evans, W., 1996: Computer-supported content analysis. Social Science Computer Review, Vol. 114, No 3: 269-279.

Fielding, N. G./Lee, R. M. (eds), 1991: Using computers in qualitative research. Thousand Oaks: Sage Publications.

Franzosi, R., 1997: Labor unrest in the Italian service sector: An application of semantic grammars. Pp. 131-145 in C.W. Roberts (ed.) (1997a).

Gerner, D. J./Schrodt, P. A./Fransisco R. A./Weedle, J. L. (1994): Machine coding of events from regional and international sources. International Studies Quarterly, 38: 91-119.

Gerner, D. J./Schrodt, P. A., 1996: The Kansas Event Data System: A beginner's guide illustrated with a study of media Fatigue in the Palestinian Intifada. Poster presented at the American Political Science Association meetings, San Fransisco, August 1996, http://wizard.ucr.edu/polmeth/working_papers96/gerner96.html.

Goldstein, J.S., 1992: A conflict-cooperation scale for WEIS events data. Journal of Conflict Resolution, 36: 369-385.

Hart, R.P., 1985: Systemic Analysis of Political Discourse: The Development of DICTION. Pp. 97-134 in Sanders et al. (eds), Political Communication Yearbook: 1984 Carbondale, IL: Southern Illinois University Press.

Huber, G.L., 1997: Analysis of Qualitative Data with AQUAD Five for Windows. Schwangau, Germany: Verlag Ingeborg Huber.

Huber, G.L./Garcia, M., 1992: Voices of beginning teachers: Computer-assisted listening to their common experiences. Pp. 139-156 in M. Schratz (ed), Qualitative voices in educational research. London: Falmer.

Kelle, U. (ed.), 1995: Computer-aided qualitative data analysis. London: Sage Publications.

Kelle, U., 1996: Computer-aided Qualitative Data Analysis: An Overview. Pp. 33-63 in C. Zuell/J. Harkness/J.H.P. Hoffmeyer-Zlotnik (eds): ZUMA-Nachrichten Spezial: Text Analysis and Computers. Mannheim: ZUMA.

Klein, H., 1997a: Classification of Text Analysis Software. Pp. 355-362 in R. Klar/O. Oppitz (eds), Classification and Knowledge Organization, Proceedings of the 20th Annual Conference of the Gesellschaft fuer Klassifikation e.V., University of Freiburg, March 6-8, 1997. Heidelberg: Springer Verlag.

Klein, H., 1997b: INTEXT-Handbuch, Version 4.0. Jena: mimeo.

Kucera, H./Francis, W. N., 1967: Computerized dictionary of present-day American English. Providence, RI: Brown University Press.

Kuckartz, U., 1998: WinMAX - Scientific Text Analyse for the Social Sciences. User's Guide. Berlin.

Kuckartz, U., 1996: MAX fuer Windows: ein Programm zur Interpretation, Klassifikation und Typenbildung. Pp. 229-243 in W. Bos/Chr. Tarnai (eds), Computerunterstuetzte Inhaltsanalyse in den Empirischen Sozialwissenschaften. Theorien - Anwendungen - Software. Muenster: Waxmann.

Leng, R. J., 1987: Behavioral Correlates of War, 1816-1975. (ICPSR 8606). Inter-University Consortium of Political and Social Research: Ann Arbor.

Lezius, W./Rapp, R./Wettler, M., 1998: A Freely Available Morphological Analyzer, Disambiguator, and Context Sensitive Lemmatizer for German. In Proceedings of the COLING-ACL 1998, Canada.

Litkowski, K. C., 1997: Desiderata for Tagging with WordNet Synsets and MCCA Categories. Proceedings of the 4th SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?, April 1997, Washington, DC.

McClelland, Ch. A., 1976: World Event/Interaction Survey Codebook. (ICPSR 5211). Inter-University Consortium of Political and Social Research: Ann Arbor.

McTavish, D. G., 1997: Scale Validity: a computer content analysis approach. Social Science Computer Review, Vol. 15, N. 4: 379-393.

McTavish, D. G./Pirro, E. B., 1990: Contextual content analysis. Quality and Quantity 24: 245-265.

McTavish, D. G./Litkowski K. C./Schrader, S., 1997: A computer content analysis approach to measuring social distance in residential organizations for older people. Social Science Computer Review, 15 (2): 170-180.

Miller, G. A./Beckwith, R./Fellbaum, Ch./Cross, D./ Miller, K./Tengi, R., 1993: Five Papers on WordNet™, CSL Report 43. Cognitive Science Laboratory. Princeton, NJ: Princeton University.

Mohler, P. Ph./Zuell, C., 1998: TEXTPACK User's Guide. Mannheim: ZUMA.

Mohler, P. Ph./Zuell, C., in print. Applied Text Theory: Quantitative Analysis of Answers to Open Ended Questions. In: M. West (ed), Progress in Communication Sciences: New directions in Computer Content Analysis.

Muhr, T., 1997: ATLAS.ti the Knowledge Workbench: Visual Qualitative Data Analysis Management Model Building. Short User's Manual (Version 4.1 for Windows 95 and Windows NT). Berlin: Scientific Software Development.

Muhr, T., 1996: Textinterpretation und Theorienentwicklung mit ATLAS/ti. Pp. 245-259 in W. Bos/Chr. Tarnai (eds), Computerunterstuetzte Inhaltsanalyse in den Empirischen Sozialwissenschaften. Muenster: Waxmann.

Ragin, C. C., 1987: The comparative method: moving beyond qualitative and quantitative strategies. Berkeley: University of California Press.

Richards, L. 1998: NUD*IST Introductory Handbook. QSR.

Roberts, C., 1989: Other than counting words: a linguistic approach to content analysis. Social Forces, 68: 147-177.

Roberts, C. W., (ed), 1997a: Text analysis for the social sciences: methods for drawing statistical inferences from texts and transcripts. Mahwah, N.J: Lawrence Erlbaum Assoc. Publishers.

Roberts, C. W., 1997b: Semantic text analysis: on the structure of linguistic ambiguity on ordinary discourse. Pp. 55-77 in C. W. Roberts (ed), 1997a.

Rostek, L., 1999: Automatische Erzeugung von semantischen Markup im Agenturmeldungen. Pp. 307-321 in W. Moehr/I. Schmidt (eds), SGML/XML – Anwendungen und Perspektiven. Heidelberg: Springer Verlag.

Rostek, L./Alexa, M., 1998: Marking up in TATOE and exporting to SGML. Computers and the Humanities, 31: 311-326.

Schmittbetz, M., 1997: Text per Mausklick am PC verstehen und ordnen. Handelsblatt, 4.03.97, Germany.

Schrodt, Ph. A., 1998: KEDS: Kansas Event Data System. Version 1.0. Kansas University. Http://www.ukansas.edu/~keds.

Schrodt, Ph. A./Gerner, D. J., 1994: Validity assessment of a machine-coded event data set for the Middle East, 1982-1992. American Journal of Political Science, 38: 825-854.

Schrodt, Ph. A./Davis, S. G./Weddle, J.L., 1994: Political Science: KEDS - A program for the machine coding of event data. Social Science Computer Review, 12, 3: 561-588.

Sperberg-McQueen, M. C./Burnard, L. (eds.), 1994: Guidelines for the encoding and interchange of machine-readable texts (TEI P3). Chicago & Oxford, April 1994.

Stone, Ph. J., 1997: Thematic text analysis: New agendas for analysing text content. Pp. 35-54 in C.W. Roberts (ed), 1997a.

Struebing, J., 1997: Computer tools for grounded theory. Pp. 399-407 in R. Klar/ O. Oppitz (eds), Classification and Knowledge Organization, Proceedings of the 20th Annual Conference of the Gesellschaft fuer Klassifikation e.V., University of Freiburg, March 6-8, 1997. Heidelberg: Springer Verlag.

Tashakkori, A./Teddlie, Ch., 1998: Mixed Methodology: combining qualitative and quantitative approaches. Thousand Oaks: Sage.

Tesch, R., 1990: Qualitative research: Analysis types and software tools. New York: Falmer.

Tesch, R., 1991: Software for qualitative researchers: analysis needs and program capabilities. Pp. 16-37 in N.G. Fielding/R.M. Lee (eds).

TextSmart™ 1.0, 1997: User's guide. Chicago, USA: SPSS Inc.

Walsh, B./Lavalli, T., 1996: Beyond beancounting: qualitative research software for business. In http://www.microtimes.com/162/research.html.

Weitzman, E. A./Miles, M. B., 1995: A software sourcebook: Computer programs for qualitative data analysis. Thousand Oaks: Sage Publications.

Zuell, C./Mohler, P. Ph. (eds), 1992: Textanalyse. Anwendungen der computer-unterstuetzten Inhaltsanalyse. Opladen: Westdeutscher Verlag.

Zuell, C./Alexa, M., 1998: Ergebnisse der TEXTPACK Nutzerbefragung. ZUMA-Technischer Bericht 15/98. ZUMA: Mannheim.